Enrico Colombini

Avventure per MS-DOS

Tutto il necessario per scrivere e giocare avventure

Edizione freeware 1999

Avendo riacquisito il copyright del volume ormai uscito di catalogo, ho deciso di autorizzarne la libera distribuzione per uso non commerciale.

Spero così di fornire qualche dettaglio in più ai numerosi giocatori che mi chiedono come si possa impostare un gioco di avventura.
Naturalmente la tecnologia software è antica (per molti aspetti lo era già alla data di uscita del libro, che deriva da pubblicazioni precedenti), ma i concetti restano validi.

A chi volesse scrivere oggi un meccanismo per avventure testo, consiglio di prendere in considerazione altri linguaggi, come il Tcl, il Perl o, al limite, Java.

Non ho modificato il contenuto, se non per alcuni aggiornamenti dei copyright, in modo da conservarne per intero il sapore archeologico.

Enrico Colombini

INDICE

[Nota: l'indice non è creato in modo automatico, per cui i numeri di pagina effettivi dipenderanno dalla vostra impaginazione; considerateli come indicativi]

Ringraziamenti:

- A Chiara Tovena, per l'insostituibile collaborazione nella stesura delle sceneggiature delle avventure, e per la consueta 'critica costruttiva'.

- A Gianni Cavallari, Roberto Cerruti, Marco Morocutti, Lucia Bonazzi, Luisa Moleri, Fulvio Francesconi, Manuela Spotti, Angelo Dolci, Riccardo Carugati, Pino Porta, Giulio Usanza, Rosy Milicia, Ruggero Valetti e vari altri amici, per i collaudi delle tre avventure.

- Al negozio "Il Computer" di Brescia per la continua disponibilità e pazienza nei confronti dei miei strani esperimenti hardware & software.

- Al mio Macintosh SE, che non mi ha mai tradito e mi ha fatto dimenticare la lentezza del glorioso Mac 128K usato per la prima edizione.

- A Roberto Pancaldi, ed a tutta la Divisione Libri Jackson [poi Jackson Libri - Gruppo Futura], per l'assistenza, la collaborazione organizzativa e la decisione di pubblicare questa seconda edizione.

BENVENUTI NEL MONDO DELL'AVVENTURA

Questo libro è dedicato a tutti gli appassionati dei giochi di avventura, ed a coloro che non lo sono ancora... ma lo diventeranno presto. Ecco una guida per orientarvi nel labirinto:

- Il Capitolo 1 spiega cosa fare del dischetto allegato alla confezione.

- Il Capitolo 2 è un'introduzione ai giochi di avventura, ed alle tecniche generali di risoluzione. Se siete già esperti, potete saltarlo.

- Il Capitolo 3 introduce le tre avventure pronte da giocare: "L'astronave condannata", "L'anello di Lucrezia Borgia" e "L'Apprendista Stregone".

- I Capitoli 4-11 vi insegnano come scrivere le vostre avventure con una conoscenza veramente minima di BASIC, usando il mio programma "Modulo Base" per liberarvi di gran parte del lavoro noioso e concentrarvi sulla trama dell'avventura.

- L'Appendice A contiene il listato del Modulo Base.

- L'Appendice B è una guida rapida di riferimento per l'uso del Modulo Base.

- L'Appendice C illustra il nuovo Modulo Base per grandi avventure.

Buona avventura!

Enrico Colombini

Nota: questo libro è stato scritto e composto direttamente con un Apple Macintosh: non contiene quindi errori tipografici o di composizione. Tutti gli errori che troverete sono garantiti originali (D.O.C.) dell'autore in persona.

CAPITOLO 1

NON È MAI TROPPO PRESTO

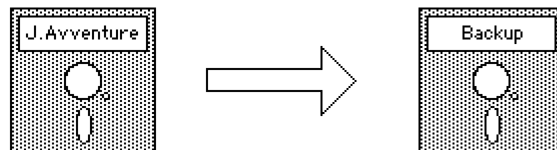
Non è mai troppo presto per salvare i vostri preziosi programmi da un triste destino. Se saltate questo capitolo, non dite poi che non vi avevo avvertito.

Il dischetto allegato alla confezione contiene le tre avventure da giocare ed il necessario per aiutarvi a scrivere le vostre avventure originali. Non correte il rischio di perdere o danneggiare il dischetto: fate subito una copia di sicurezza (altrimenti detta backup).

Per fare la copia di backup, avviate il calcolatore con un disco contenente il sistema operativo MS-DOS (versione 2.0 o successiva) ed il programma DISKCOPY.COM, tenete un dischetto vuoto a portata di mano e scrivete:

`DISKCOPY A: A:` (`DISKCOPY A: B:` se avete due floppy disk drive)

Inserite poi il dischetto originale (sorgente) e quello vuoto (destinazione) quando il programma li richiede.
In ogni caso, fate attenzione a non cancellare per errore il dischetto originale (specialmente se avete un solo disk drive). Per sicurezza, mettetegli l'apposita etichetta di protezione dalla scrittura (se non c'è già) prima di iniziare la copia, così il disk drive non può scrivere sul disco originale nemmeno se vi sbagliate.



Una volta terminata la copia, mettete via l'originale in un posto sicuro e lavorate con la copia. È meglio usare la copia, dato che è stata registrata sul vostro disk drive ed è quindi improbabile che possa darvi problemi, anche in caso di regolazione non perfetta del drive stesso.
Non tenete originale e copia nella stessa scatola, altrimenti metà della sicurezza data dal backup è persa in partenza. Conosco un esperto programmatore che, in un raptus di distrazione, ha usato una scatola di dischetti come supporto per un altoparlante: ha danneggiato i dati su quasi tutti i dischetti contenuti nella scatola! Questo per dire che l'imprevisto colpisce nei modi più strani. Se no, che imprevisto sarebbe?

Le vostre avventure

Siete comunque liberi di usare i programmi allegati a questo libro per produrre avventure commerciali senza dover pagare alcun diritto, purché ci sia una citazione "Costruito con il Modulo Base di Enrico Colombini", o equivalente.

Note [le lascio per motivi storici]:

- Per motivi di copyright, il dischetto fornito non contiene MS-DOS ed interprete BASIC. Non è quindi possibile usarlo come disco di bootstrap, cioè per far partire il calcolatore direttamente all'accensione.

- Il dischetto da 5" è registrato nel comune formato da 360K (40 tracce, 2 facce), in modo da poter essere leggibile dalla grande maggioranza dei disk drive usati su computer MS-DOS. I file possono essere trasferiti senza problemi su dischi formattati con capacità maggiore (come quelli del PC-AT), su dischi di diverso formato (es. 3.5") o su hard disk.

CAPITOLO 2

ALL'AVVENTURA!

Raccogliete le vostre cose e salutate gli amici: è giunta l'ora di partire per un mondo sconosciuto, dove vi aspettano gloria e fortuna. Oppure una fine solitaria ed ingloriosa. A voi la scelta!

Il fascino dei giochi di avventura sta nel loro potere: quello di creare un'illusione, facendo vivere il giocatore in un mondo immaginario, ma quanto mai reale per chi vi è immerso. Intendiamoci, non tutte le ciambelle riescono col buco: come ci sono romanzi avvincenti e polpettoni da quattro soldi, così si trovano avventure affascinanti ed altre semplicemente noiose. Riprenderò il discorso nei capitoli dedicati alla creazione delle vostre avventure.

L'avventura attrae (anche, ma non solo) per lo stesso motivo che rende interessanti rebus ed enigmi: la soddisfazione che si prova quando si riesce a risolverli, e che è tanto maggiore quanto più il problema era apparentemente insolubile. Anche qui, non basta che il problema sia difficile: deve essere alla portata del giocatore, altrimenti c'è solo frustrazione invece che divertimento.

Lo schema di un gioco di avventura è semplice: il calcolatore descrive (o disegna) il luogo in cui l'avventuriero si trova in quel momento. Il giocatore (che comanda l'avventuriero e con lui si identifica) può fare una serie di azioni, semplicemente scrivendo comandi alla tastiera. Per esempio:

`APRI LA PORTA`

e riceve in risposta la descrizione delle conseguenze. Ovviamente, la libertà di azione non è illimitata. Tutti i programmi di avventura 'capiscono' almeno le direzioni, ad esempio:

`NORD` (o semplicemente `N`)

significa "spostati nel luogo adiacente a Nord di quello in cui ti trovi", e così pure per le altre direzioni (non dimenticando `ALTO` e `BASSO`).
Vagando per le stanze (o genericamente "luoghi"), capita di trovare degli oggetti che possono essere utili. In questo caso, basta dire:

`PRENDI IL PUGNALE`

per appropriarsi dell'oggetto in questione, e:
`LASCIA IL PUGNALE`

per abbandonarlo nel luogo in cui ci si trova.
Un'altra azione di fondamentale importanza è `GUARDA`, che fornisce quasi sempre utili informazioni:

`GUARDA IL FORZIERE`

potrebbe dare come risposta: "E' aperto e contiene il tesoro che cercavi" (anche se è molto improbabile che sia così facile...).
Se si incontrano altre persone, o comunque esseri più o meno intelligenti, può essere utile anche una conversazione:

```
PARLA CON IL DIAVOLO
```

che, magari, ha giusto un contratto da proporre (io non mi fiderei troppo...).
C'è sempre modo di conoscere gli oggetti che si possiedono in un dato momento, ad esempio scrivendo:

```
INVENTARIO
```

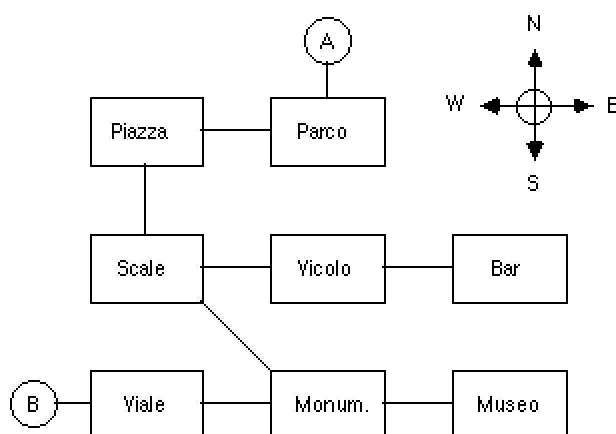
Di solito, è anche possibile registrare la situazione di gioco per riprenderla in un momento successivo, o per ripartire da un punto noto in caso di spiacevoli sorprese (ritrovarsi a cena con un drago, nel ruolo dell'antipasto, o cose del genere), con appositi comandi del tipo:

`SALVA` (per registrare) e
`RIPRENDI` (per ripartire dal punto dove avevate registrato)

Questi comandi, come dicevo, sono presenti in tutti i giochi di avventura (le parole di comando non sono necessariamente le stesse). Le altre azioni ammesse sono lasciate alla fantasia dell'autore, ed alla qualità del programma. Le avventure meno interessanti sono quelle che ammettono una sola azione per risolvere un certo problema, e rispondono "Non capisco" in tutti gli altri casi. Le migliori sono ricche di risposte non direttamente utili alla soluzione del problema, ma che contengono prese in giro, indizi, o contributi al realismo dell'ambientazione.

Come si risolve un'avventura

La prima cosa da fare è tracciare una mappa quanto più possibile accurata, provando in ciascun luogo le sei possibili direzioni. Io uso di solito una tecnica di questo genere: rappresento ogni luogo con un rettangolo, collegato ai luoghi adiacenti con una linea, così:



Dove non è possibile andare, si può sbarrare la direzione con un trattino, in modo da ricordarsi le direzioni provate, e non dimenticare di provarle tutte. Le linee diagonali, che partono dagli spigoli superiori od inferiori, indicano rispettivamente movimento verso l'alto e verso il basso (ad esempio: dalle scale si scende verso il monumento). Le lettere nei cerchiolini sono rinvii ad altri fogli dove prosegue la mappa, se non ci sta tutta su un foglio solo.

È fondamentale provare tutte le direzioni in tutti i luoghi, altrimenti si rischia di restare bloccati in situazioni senza uscita, quando magari ad Est si stende un vasto territorio inesplorato. In casi disperati, può valer la pena di ricontrollare tutta la mappa alla ricerca di eventuali errori.

Una volta determinata la geografia dei luoghi (o almeno di quelli immediatamente raggiungibili), conviene osservare per bene tutti gli oggetti ed i dettagli che sembrano interessanti (con `GUARDA`). È un'altra operazione essenziale, se non si vuole correre il rischio di perdere informazioni vitali.

Terminato questo lavoro preliminare, si può cominciare a raccogliere oggetti (prendendo nota sul solito foglio) ed a provare azioni di ogni genere. Ma attenzione: prima di comportarsi in modo sconsiderato, o comunque di compiere operazioni rischiose (come `UCCIDI IL DRAGO`), conviene registrare la situazione (con `SALVA`) per non dover ricominciare daccapo in caso di (probabile) esito disastroso.

Qualche altro consiglio, in ordine sparso:

- Se possibile, non giocate da soli. In compagnia ci si diverte molto di più ed è più facile avere buone idee.

- Leggete sempre le risposte con molta attenzione: spesso contengono indizi o informazioni essenziali, che possono sfuggire ad un lettore frettoloso.

- Provate tutto, ma proprio <u>tutto</u> quello che vi salta in mente, anche se vi sembra del tutto stupido, inutile od assurdo. Non è mai detto.

- Cercate di entrare nello spirito del gioco: alcune avventure richiedono forza bruta (`ATTACCA IL DRAGO`), altre giocano sull'astuzia o il ragionamento (`PARLA CON IL DRAGO`), altre sono surreali od originali (`ACCAREZZA IL DRAGO`).

- Tornate più volte negli stessi posti: la scena può essere cambiata (di solito c'è qualche indizio che lo suggerisce).

- Se una parola non viene accettata (ad esempio `ROMPI`), provate qualche sinonimo (come `SPACCA`, `SPEZZA`, `FRANTUMA`, `DISTRUGGI`) prima di concludere che l'azione non è quella giusta.

Ed ora, è giunto il momento di affrontare la prima avventura!

CAPITOLO 3

TECNOLOGIA, CAVALLERIA, STREGONERIA

Tre avventure da giocare: la prima ambientata su un'astronave di linea che solo la vostra abilità può salvare da una sicura catastrofe, la seconda in pieno sedicesimo secolo dove si dimostrerà se siete degni del titolo di cavaliere che portate, e la terza in un reame dove la magia non è mai tramontata.

Le tre avventure contenute nel dischetto sono alquanto diverse tra loro, e non solo nell'ambientazione. Vi consiglio di giocarle nell'ordine, dato che il livello di complessità cresce dalla prima alla terza.

L'astronave condannata è nata soprattutto per illustrare l'impiego del programma per la scrittura di semplici avventure in BASIC, ma si tratta comunque di un'avventura completa e giocabile. Gli avventurieri in erba vi troveranno la loro prima sfida, gli esperti possono usarla per scaldare i muscoli in vista delle successive (ma non è detto che la risolvano in dieci minuti!).
Per far partire il gioco, occorre innanzitutto avviare il calcolatore con il sistema operativo MS-DOS (versione 2.0 o successiva). Poi si deve far partire il BASIC con il comando:

`GWBASIC` (per la maggior parte dei compatibili)

oppure:

`BASICA` (per gli IBM)

Se l'interprete GWBASIC non è fornito con il computer, il rivenditore che vi ha venduto la macchina sarà certamente in grado di procurarvelo.

A questo punto, togliete il dischetto del sistema operativo (se non avete un hard disk) ed inserite la copia del disco fornito. Se non avete fatto la copia come descritto nel Capitolo 1, è ora di farlo. Se infatti inserite l'originale, questo si cancella (non è vero, ma ve lo meritereste...).
Finalmente, potete far partire il programma, assicurandovi che Caps Lock (Blocco Maiuscole) sia premuto e scrivendo:

```
LOAD "ASTRO",R
```

Dopo qualche secondo, sarete a bordo (e nei guai).
Le azioni fondamentali sono quelle citate nel capitolo 2. Ve le riassumo:

`N, S, E, O, A, B`	sono le sei possibili direzioni.
`PRENDI` (qualcosa)	e
`LASCIA` (qualcosa)	servono per manipolare gli oggetti.
`GUARDA` (qualcosa)	fornisce utili indizi.

`SALVA o SAVE`	registra la situazione corrente.
`RIPRENDI o LOAD`	riprende la situazione registrata.
`INVENTARIO`, `COSA`, o semplicemente '?'	elenca gli oggetti posseduti.

Il programma capisce anche altre parole, che vi appariranno (spero) ovvie nelle varie situazioni. Se proprio non ne venite fuori, potete alzare bandiera bianca (buu!) e dare una sbirciatina alle linee di programma dalla 40000 in poi. Chissà che non vi suggeriscano qualcosa.

La seconda avventura, L'anello di Lucrezia Borgia, è molto più complessa e rifinita. Pur essendo ben lontana dal livello di dettaglio di "Avventura nel castello", sono pronto a scommettere che darà filo da torcere anche agli avventurieri più smaliziati. Per farla partire (sempre da BASIC), scrivete:

```
LOAD "LUCREZIA",R
```

Dato che incontrerete varie persone, mi sembra il caso di rivolgere loro la parola: potrebbero darvi utili informazioni. Inoltre, ecco un prezioso consiglio: provate tutte le strade, e non solo in senso geografico. Non è detto che siano vicoli ciechi.

Una volta capito lo spirito dell'avventura, le cose dovrebbero essere molto più facili. Se siete abbastanza svegli, naturalmente.
La terza avventura, L'Apprendista Stregone, è una 'grande' avventura che mostra le notevoli possibilità del nuovo Modulo Base, descritto nell'Appendice C per i lettori più esperti. Per farla partire, scrivete (da BASIC):

```
LOAD "STREGONE",R
```

Questa volta siete sufficientemente esperti da non aver bisogno di suggerimenti e spunti.

CAPITOLO 4

CREARE UN'AVVENTURA

Non è necessario essere esperti programmatori per scrivere un gioco di avventura: con l'aiuto del Modulo Base, basta una conoscenza elementare del BASIC.

Dopo aver giocato con "L'astronave condannata", "L'anello di Lucrezia Borgia" e "L'Apprendista Stregone", sono sicuro che vi sarà venuta voglia di scrivere una vostra avventura. Come si fa? Da che parte si comincia? Aiuto!
Calma e gesso, come dicono i giocatori di bigliardo. Il lavoro si divide in due parti:

- Ideare la trama dell'avventura.
- Scrivere il programma che la realizza.

La prima è la parte creativa, la seconda richiede soltanto tecnica. È un po' come dipingere un quadro: non basta avere l'immagine in mente, bisogna anche trasferirla sulla tela.
È chiaro che le maggiori difficoltà vengono nella seconda parte: la scrittura del programma. Ci sono in circolazione, è vero, programmi già fatti che consentono di scrivere semplici avventure con poca fatica, ma obbligano l'autore a restare in uno schema predefinito. Inoltre, lasciano completamente all'oscuro del funzionamento interno del programma: funziona, e non chiedetevi perché.
Ho preferito, invece, lavorare per due obbiettivi:

- Sollevare l'autore di avventure dal lavoro tecnico e noioso.
- Mostrare come funziona un gioco di questo tipo.

Ne è venuto fuori un programma che ho chiamato "Modulo Base" (una versione semplificata dell'ADL-1 che ho usato per "Avventura nel Castello"), con il quale è possibile scrivere avventure con un minimo di conoscenza di BASIC, e con la massima libertà di introdurre varianti ed effetti speciali. Per mostrare che non c'è nulla di magico o misterioso in un programma del genere, l'ho scritto in BASIC "pulito", senza trucchi o routine in assembly.
Ovviamente, "L'astronave condannata" e "L'anello di Lucrezia Borgia" sono scritte impiegando il Modulo Base. "L'Apprendista Stregone" è scritta invece con il nuovo Modulo Base potenziato, descritto nell'Appendice C.
Il progetto ed il meccanismo di funzionamento del Modulo Base sono esaminati in dettaglio nel volume "Scrivere un'avventura" (stesso autore,ed.Jackson). In queste pagine vi guiderò invece alla realizzazione di una semplice avventura di esempio, per illustrarvi come si usa il programma.

Progettare un'avventura

Il progetto di un'avventura richiede prima di tutto un'ispirazione, cioè l'idea base della trama, l'ambientazione e lo schema generale della vicenda.

Una volta che si ha ben chiaro in mente quello che si intende fare, bisogna passare al dettaglio. Per poterla trasformare in un programma, la trama dev'essere accuratamente definita in ogni particolare:

- La mappa dei luoghi in cui si svolge l'avventura.
- Il dizionario, cioè l'elenco delle parole conosciute dal programma.

- Gli oggetti ed i personaggi che l'avventuriero può incontrare.
- Le azioni, cioè le frasi che hanno effetto sul gioco nelle varie situazioni.

Prima di iniziare a scrivere un'avventura, bisogna preparare un dischetto di lavoro. Prendete un dischetto nuovo, formattatelo e copiatevi sopra il Modulo Base.
Se avete poca esperienza di DOS e non vi sentite troppo sicuri (o avete un solo drive), ripetete invece il procedimento descritto nel capitolo 1 e fate una seconda copia del dischetto originale, dalla quale cancellate tutti i programmi eccetto `BASE.BAS`. Ad esempio:

`DEL LUCREZIA.BAS` (solo sulla copia di lavoro, NON fatelo sull'originale!!!)

e così via per gli altri programmi. Alla fine, avrete un dischetto che contiene solo il programma `BASE.BAS`, che è per l'appunto il Modulo Base. È anche consigliabile preparare un ulteriore dischetto inizializzato, da usare per le copie di sicurezza del lavoro in corso.

Vi sembra che ci siano troppe copie? Non si può lavorare direttamente sul dischetto originale? Se fate queste domande, avete ancora molta strada da fare prima di diventare programmatori professionisti. Lavorare senza backup (copie di riserva) ricorda Chernobyl: è molto improbabile che succedano incidenti, ma se avvengono...

CAPITOLO 5

OPERAZIONE ZANNA BIANCA

Prima di cominciare a lavorare sul programma BASIC bisogna impostare, almeno nelle linee generali, la trama dell'avventura che si vuol realizzare.

Non commettete anche voi l'errore di mettervi a scrivere un'avventura senza aver ben chiara in mente (o, meglio ancora, su carta) la trama ed i dettagli. Risparmiereste, sì, un poco di lavoro all'inizio, ma finireste col pagarlo caro. Andando avanti nella scrittura del gioco, finireste col trovarvi persi in una miriade di dettagli che non collimano, come in un giallo scritto male. Meglio dunque perdere qualche ora per progettare l'avventura nei minimi particolari.

Tanto per smentirmi subito, non seguirò il mio stesso consiglio, e costruirò un esempio un passo alla volta, aggiungendo man mano nuovi dettagli. Sia chiaro che lo faccio soltanto per non presentarvi le difficoltà tutte insieme. Inoltre, l'esempio è veramente molto semplice. E poi, me lo sono preparato nei dettagli prima di iniziare a scrivere.

La micro-avventura che userò per illustrarvi come si lavora col Modulo Base si chiama "Operazione Zanna Bianca". È la vicenda di un agente segreto, che deve 'prelevare' con destrezza dei microfilm riguardanti le rotte polari dei sommergibili nucleari (non si dice di quale Potenza: scegliete a vostro piacimento).

È chiaro che non si tratta di una vera avventura, ma solo di un esempio che, tuttavia, contiene quasi tutti gli elementi usabili per la costruzione di avventure vere e proprie. Non scandalizzatevi dunque per la semplicità e la mancanza di dettaglio. Anzi, se volete poi trasformare l'esempio in una vera avventura, la strada è aperta per la vostra fantasia.

Prima di passare alla parte tecnica, due parole sulla creazione delle avventure. Ci sono alcuni punti da tenere ben presenti, se non volete semplicemente aggiungere un'altra avventura insulsa alle troppe che già circolano. Cercherò di indicarvi i principali:

- La trama dev'essere coerente. Le incoerenze spezzano il filo della vicenda, e rompono il fragile incantesimo che tiene avvinto il lettore/giocatore (ci sono molti punti in comune tra un'avventura ed un racconto). Parole magiche non hanno senso a bordo di un'astronave, come un tesoro stona in un'avventura di spionaggio: fa cadere la tensione e distrae il giocatore dallo scopo principale.

- L'atmosfera è importante, per fare in modo che il giocatore si immedesimi col personaggio. Anche se il Modulo Base (nella sua versione più semplice) non permette lunghe descrizioni (possibili invece con il nuovo Modulo Base, vedi Appendice C), basta spesso un aggettivo per stimolare la fantasia del giocatore. "Una sala" è semplicemente una sala, ma "Una sala silenziosa" è tutt'altra cosa: chi immagina la polvere dei secoli depositata sugli antichi arredi, chi sospetta che qualcosa si celi dietro il silenzio innaturale... In ogni caso, non è un posto qualunque (anche se, magari, non c'è proprio niente di utile ai fini del gioco).

- Lo scopo dell'avventura è il divertimento dei giocatori, non il sadico compiacimento dell'autore che nessuno riesca a risolverla. I problemi devono essere abbastanza difficili da dare soddisfazione quando li si risolve, ma non impossibili: altrimenti non ci si diverte. Bisogna collaudare l'avventura, cioè farla giocare a vari amici: se troppo pochi riescono a risolvere un certo problema, è il caso di fornire qualche indizio in più.

- Evitare l'errore più grave: che le difficoltà siano risolubili con una sola azione, descritta con una sola frase valida. I collaudi servono soprattutto a questo. Se avete previsto l'azione "Rispondi al telefono", troverete che molti scrivono invece "Prendi il telefono" o "Alza la cornetta" o simili. Se ottengono solo risposte del tipo "Non capisco" o, peggio, "Non succede niente", probabilmente dopo un po' lasciano perdere.

- Sono i dettagli che rendono interessante un'avventura, molto più che la complessità della vicenda. Più risposte inutili o secondarie sono state previste, più il gioco è appassionante ed i giocatori si divertono. Per esempio, in risposta ad "Accendi il televisore", una risposta tipo "Stanno trasmettendo la 3257º puntata di Dallas" è senz'altro meglio del solito "Non capisco".

Anche qui, i collaudi sono molto importanti: prendete nota di tutto quello che scrivono i vostri amici, ed aggiungete risposte adeguate. I giocatori non riusciranno a staccarsi dal computer, e si chiederanno come diavolo avete fa il programma a 'capire' quello che scrivono alla tastiera, non sapendo che siete stati voi a prevedere tutte le azioni che loro stanno facendo.

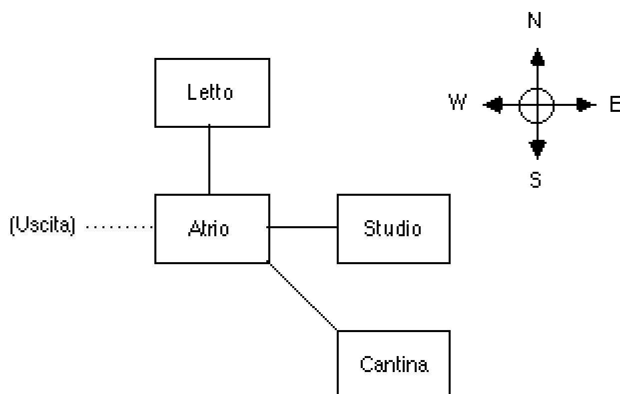
CAPITOLO 6

LA CASA DELL'AMBASCIATORE

Il programma si occupa automaticamente degli spostamenti. Tutto quello che c'è da fare è introdurre le descrizioni dei luoghi ed i loro collegamenti nelle apposite linee di DATA del Modulo Base.

Ovviamente, i microfilm si trovano nell'appartamento dell'ambasciatore. Per semplicità, tralascio tutte le stanze accessorie (bagno, cucina, ecc.), e considero soltanto quelle che hanno a che fare con l'avventura.

Questa è la mappa stilizzata dell'appartamento:



Come si vede, dall'atrio si può andare a Nord (camera da letto), ad Est (Studio) ed in basso (cantina). Si potrà anche andare ad Ovest per uscire dal gioco, ma per ora non ci interessa (non esiste, in effetti, un altro luogo ad Ovest).
Elenchiamo per esteso i vari luoghi possibili, numerandoli a partire da uno ed indicando i vari collegamenti tra un luogo e l'altro:

Luogo:	Nord	Sud	Est	Ovest	Alto	Basso
1) Camera da letto	-	atrio	-	-	-	-
2) Atrio	letto	-	studio	-	-	cantina
3) Studio	-	-	-	atrio	-	-
4) Cantina	-	-	-	-	atrio	-

I numeri dei luoghi sono importanti: nel programma sono l'unico modo per distinguere un luogo dall'altro.

La camera da letto è il numero 1, l'atrio è il numero 2, lo studio è il numero 3 e la cantina, infine, è il numero 4. Anzi, vale la pena di riscrivere la tabella mettendo in evidenza i numeri (anziché i nomi) dei vari luoghi:

Luogo:	Nord	Sud	Est	Ovest	Alto	Basso
1) Camera da letto	-	2	-	-	-	-
2) Atrio	1	-	3	-	-	4
3) Studio	-	-	-	2	-	-
4) Cantina	-	-	-	-	2	-

Questa tabella rappresenta la mappa della nostra avventura. Ci sono infatti tutte le informazioni necessarie per determinare l'effetto di uno spostamento.

Ad esempio, muovendosi in alto dal luogo 4 (cantina) si finisce nel luogo 2 (atrio), mentre uno spostamento in basso dal medesimo luogo 4 (cantina) non è possibile.

Ora che la tabella è pronta, non resta che introdurla nel Modulo Base.

Introdurre la mappa

Per lavorare sul Modulo Base, occorre caricarlo. Dunque, fate partire il BASIC (GWBASIC o BASICA) e scrivete:

```
LOAD "BASE"
```

È il caso di cambiare subito le prime tre linee di REM:

```
100 REM ## Avventura: Interprete e modulo base ##
110 REM ##         di Enrico Colombini         ##
```

sostituendole con qualcosa del genere:

```
100 REM ## Avventura: Operazione Zanna Bianca ##
110 REM ##          di Pinco Pallino          ##
120 REM ## su Modulo Base di Enrico Colombini ##
```

Compiuta questa importante formalità, possiamo passare a qualcosa di più sostanzioso. Osservate le linee:

```
44970 REM
44980 REM - MAPPA:
44990 REM
45000 DATA "FM": REM --- LUOGHI E MAPPA, DAL LUOGO N.1 ---
```

È proprio qui, a partire dalla linea 45000, che vanno inserite le caratteristiche dei luoghi, in ordine di numero. Il primo elencato sarà il numero 1, il secondo il numero 2, e così via. Iniziamo dunque dal numero 1:

```
45000 DATA "in un'ampia camera da letto","000200000000"
```

Cosa significa tutto questo? Procediamo con ordine. Ogni linea deve iniziare con il numero di linea, seguito dall'istruzione DATA. Per comodità, conviene usare

numeri di linea di 10 in 10. La prima linea di DATA della mappa deve essere la numero 45000.

Poi va messa la descrizione del luogo (tra virgolette), considerando che verrà stampata preceduta da "Sei " e seguita da un punto. Il luogo numero 1 appena introdotto verrà dunque stampato così:

Sei in un'ampia camera da letto.

Infine, va messo l'elenco dei luoghi adiacenti. Ricordate la tabella dei collegamenti? Questa era la linea riguardante la camera da letto:

1) Camera da letto	-	2	-	-	-	-

Riscriviamola usando solo numeri di due cifre, dunque 02 invece di 2, ed indicando con 00 le direzioni non ammesse (dove non si può andare):

1) Camera da letto	00	02	00	00	00	00

Scriviamo ora di seguito i sei gruppi di due cifre (riguardanti nell'ordine: Nord, Sud, Est, Ovest, Alto, Basso), ottenendo una stringa di 12 cifre:

000200000000

Questa stringa va riportata (tra virgolette) dopo la descrizione del luogo, separata da questa con una virgola, ottenendo per l'appunto:

```
45000 DATA "in un'ampia camera da letto","000200000000"
```

A stretto rigore, le virgolette non sono obbligatorie. Se non le mettete, fate però attenzione: non potete introdurre caratteri come virgola, due punti e punto e virgola nella descrizione del luogo, ed eventuali spazi in più possono creare problemi. A scanso di guai, è meglio metterle sempre.

La lunghezza massima della descrizione è limitata dalla lunghezza di una linea di programma.

Diciamo che potete arrivare a poco più di 200 caratteri, anche se è consigliabile non sprecare troppa memoria per le descrizioni (ce n'è sempre troppo poca).

Il nuovo Modulo Base (vedi Appendice C) consente di leggere da disco descrizioni di dimensioni illimitate.

Sistemato il primo luogo, ripetiamo il medesimo procedimento per i successivi. Alla fine, questo è il risultato:

```
45000 DATA "in un'ampia camera da letto","000200000000"
45010 DATA "in un atrio arredato con stile","010003000004"
45020 DATA "nello studio dell'Ambasciatore","000000020000"
45030 DATA "in una cantina ammuffita","000000000200"
45040 DATA "FM"
```

Queste linee comunicano al programma tutto quanto è necessario sapere sui luoghi dove si svolge l'avventura e la loro disposizione. Fate molta attenzione a non dimenticare la linea finale:

```
45040 DATA "FM
```

altrimenti il programma non sa quando i luoghi sono finiti e si ferma con errore prima ancora di partire (indicando un errore intorno alla linea 1100), o comunque si confonde parecchio.

A questo punto, è il caso di salvare il programma prima di provarlo:

```
SAVE "ZANNA1"
```

Verifica e collaudo della mappa:

Per controllare il lavoro fatto fino a questo momento, fate partire il programma :

```
RUN
```

Apparirà la scritta:

`Un attimo di pazienza...` (il programma si legge i DATA)

poi lo schermo verrà cancellato ed apparirà:

`- Premi <spazio> per continuare -`

Se invece vi capita un:

`?SYNTAX ERROR`

oppure un:

`?OUT OF DATA ERROR`

o altre spiacevolezze del genere, significa che avete sbagliato ad introdurre i luoghi: verificate bene che ogni linea inizi con DATA, seguito dalla descrizione del luogo (tra virgolette), poi una virgola e le 12 cifre dei collegamenti (sempre tra virgolette, e controllate che siano proprio 12).

Controllate infine he ci sia "`FM`" alla fine (maiuscolo e tra virgolette).

Se va tutto bene, obbedite al programma e premete la barra spaziatrice. Leggerete sul video:

`Sei in un'ampia camera da letto.`

`Cosa devo fare?`
Già, cosa dovete fare? Per ora, dovete provare a spostarvi nelle varie direzioni, per verificare che la mappa sia corretta. Rispondete dunque:

`Cosa devo fare? NORD`

(o semplicemente **`N`**, comunque maiuscolo). Il programma risponde:

`- Di li' non puoi andare.`

e questo corrisponde con la nostra mappa. Scrivendo invece:

`Cosa devo fare? SUD`

succede che:

Sei in un atrio arredato con stile.

Cosa devo fare?

Fin qui, tutto bene. Ora, c'è da fare un lavoro noioso ma necessario: provare tutte le sei possibili direzioni (N/S/E/O/A/B) in tutti i luoghi, per verificare che la mappa sia corretta in ogni particolare. Se trovate qualche errore, correggetelo nella corrispondente linea di DATA.

> La prova esaustiva (provare tutte le possibilità) è noiosa ma è l'unica garanzia che non ci siano fastidiosi errori nascosti.

In alcune avventure, può darsi che vi siano luoghi non direttamente collegati tra loro: ad esempio, luoghi in cui si arriva prendendo un aereo, o trasportati da una parola magica.

Come fare a controllare anche questi, se non ci si può ancora arrivare? Vi insegno un trucco.

Supponiamo che lo studio non sia direttamente raggiungibile dall'atrio, o comunque dal luogo in cui vi trovate. Per andarci, quando il programma scrive:

Cosa devo fare?

fate CTRL-C (tenendo premuto il tasto CTRL, battete brevemente una C). Il programma, colpito alla schiena, si arresterà bruscamente lamentando un:
Break in 800

Ora il programma è fermo. Se lo desiderate, potete anche listare (LIST) una o più linee, ma fate attenzione a non modificare alcuna linea, altrimenti perdete tutte le variabili, cioè le informazioni che il programma si era letto dai DATA o creato per i fatti suoi.

Per andare nello studio, che è il luogo 3 (terza linea dei DATA della mappa), scrivete:

LU=3: GOTO 760

Otterrete:

Sei nello studio dell'ambasciatore.

Siete arrivati direttamente alla vostra destinazione: il luogo 3, cioè lo studio. Questo sistema, come vedremo, servirà molto nella messa a punto del gioco.

Tecnicamente, avete fermato il programma (CTRL-C), cambiato il valore di una variabile (`LU=3`), e fatto ripartire il programma dall'inizio del ciclo di gioco (`GOTO 760`).

Per gli esperti: si potrebbe usare anche l'istruzione CONT per far ripartire il programma dalla INPUT, ma non si avrebbe la descrizione del nuovo luogo.

Il programma può essere arrestato anche con CTRL-Break (anziché CTRL-C), con la differenza che CTRL-C agisce solo quando il programma è in input (aspetta cioè una frase da tastiera), mentre CTRL-Break può fermare il programma in qualunque punto ed è dunque da usare con maggior cautela.

Tornando al punto principale:

La variabile **LU** contiene il numero del **luogo corrente**, cioè quello in cui si trova l'avventuriero. Cambiando il numero contenuto in questa variabile, si cambia il luogo dell'avventuriero.

A proposito, quando volete fermare il programma per lavorarci sopra, fate CTRL-C come prima. Per ora, non c'è modo più 'pulito' per fermarlo.

(In un'avventura, sarebbe bene consentire al giocatore di smettere scrivendo BASTA, FINE, o parola equivalente).

Introduzione all'avventura

Avrete notato, durante il collaudo, che il gioco inizia in camera da letto. Perché proprio lì? Perché la camera da letto è il luogo numero 1, essendo il primo nell'elenco dei DATA.

Non è però obbligatorio partire dal luogo 1, anzi è possibile far iniziare l'avventura nel luogo preferito. Nel nostro caso, direi che il luogo più indicato è l'atrio, cioè il luogo numero 2.

C'è anche un'altra cosa da fare: scrivere una breve presentazione ed introduzione all'avventura, che venga stampata ogni volta che si inizia il gioco.

Entrambe queste operazioni sono compito di un'apposita routine di introduzione al gioco, che si trova alla linea 4500.

Al momento, è alquanto scarna:

```
4480 REM - INTRODUZIONE -
4490 REM
4500 CLS:PRINT:PRINT:PRINT
4510 REM --- METTERE QUI INTRODUZIONE AVVENTURA ---
4520 REM --- RICORDARSI LU=LUOGO INIZIALE ---
4950 LU=1:PRINT:RETURN
```

Notate che la linea 4950 contiene **LU=1**.

È questo che fa in modo che il gioco inizi nel luogo 1. Basta una piccola modifica:

```
4950 LU=2:PRINT:RETURN
```

E, facendo RUN:

dopo il solito:

- Premi <spazio> per continuare -

il gioco inizia con:

Sei in un atrio arredato con stile.

Risolto il problema di far iniziare il gioco nel luogo voluto, possiamo scrivere l'introduzione che il giocatore leggerà all'inizio dell'avventura.

Si tratta soltanto di mettere alcune istruzioni PRINT, a partire dalla linea 4510:

```
4510 PRINT "*** Operazione Zanna Bianca ***"
4520 PRINT:PRINT
4530 PRINT "Questa e' la tua missione:"
4540 PRINT "Recuperare i microfilm o morire!"
```

Quello che va stampato dev'essere tra virgolette, e preceduto dall'istruzione PRINT. Le due PRINT a vuoto alla linea 4520 servono per stampare due linee vuote di stacco. Nel complesso, la routine di introduzione deve presentarsi così:

```
4480 REM - INTRODUZIONE -
4490 REM
4500 CLS:PRINT:PRINT:PRINT
4510 PRINT "*** Operazione Zanna Bianca ***"
4520 PRINT:PRINT
4530 PRINT "Questa e' la tua missione:"
4540 PRINT "Recuperare i microfilm o morire!"
4950 LU=2:PRINT:RETURN
```

Gli esperti potranno notare che ci starebbe tutto su una sola linea, separando le varie istruzioni con due punti. Ho preferito la chiarezza, anche se occupa un poco di spazio in più.

Ora, registrate il programma su disco:

```
SAVE "ZANNA2"
```

Perché "ZANNA2" e non "ZANNA1"? Perché non conviene salvare sempre con lo stesso nome, altrimenti un qualunque problema (ad esempio una mancanza di corrente, o lo sportello del disk drive chiuso male) causa la perdita di tutto il lavoro precedente. Inoltre, è possibile risalire alle versioni precedenti in caso di modifiche errate.

Ed ora, il solito RUN di prova:

Un attimo di pazienza...

e poi:

***** Operazione Zanna Bianca *****

Questa e' la tua missione:
Recuperare i microfilm o morire!

- Premi <spazio> per continuare -
Naturalmente ci vorrebbe un'introduzione più sostanziosa, ma questo è solo un esempio. Potete scrivere tutto quello che vi pare.

La scritta:

- Premi <spazio> per continuare -

appare automaticamente alla fine dell'introduzione. Se volete farla apparire più volte, ad esempio perché l'introduzione non ci sta in un solo schermo, usate l'istruzione:

```
GOSUB 1500
```

per fermare la stampa tra una PRINT e la successiva.

Non mettete pause (ritardi) nell'introduzione, sono molto fastidiose (anzi, per ora non vi spiego nemmeno come si fa).

È tutto chiaro?

Spero di sì. Non vi conviene andare avanti senza aver assimilato, e provato in pratica, il contenuto di questo capitolo.

Un riassunto assai sintetico di quello che ho finora illustrato, utile per non dover cercare informazioni diluite in varie pagine, è riportato nell'Appendice B.

CAPITOLO 7

CHIAVE, PORTA E MICROFILM

In un'avventura si incontrano oggetti di vario tipo. Per fare in modo che siano trattati dal programma in modo automatico, basta introdurre il loro nome nel dizionario e le loro caratteristiche nell'apposito elenco.

Per avere un'avventura, non basta una mappa. Bisogna anche che ci siano dei problemi da risolvere. Di solito, si tratta di azioni da compiere su oggetti, per esempio:

```
APRI LA PORTA
```

Perché questo sia possibile, occorre che:

- Il programma 'capisca' le parole APRI e PORTA.
- Ci sia un oggetto PORTA su cui agire.
- L'azione APRI LA PORTA sia stata prevista.

Partiamo dal primo punto: se vogliamo che il programma riconosca le parole che scriviamo, bisogna che in qualche modo le conosca.

Prima ancora, bisogna che noi ci prepariamo un elenco delle parole che potranno servire nel corso dell'avventura. Anche se è possibile aggiungerle una alla volta, è più comodo pensarne quante più possibile ed introdurle in un colpo solo.

Per decidere quali parole insegnare al programma, dobbiamo avere un'idea di quello che può succedere durante il gioco.

Per non complicare troppo le cose, schematizziamo l'avventura in questo modo:

- I microfilm sono nello studio.
- La porta che dall'atrio dà nello studio è chiusa a chiave.
- La chiave si trova in cantina.

Ora, scriviamo la sequenza principale, cioè la serie di azioni che porta alla soluzione del gioco, a partire dall'atrio:

B (o BASSO)	(va in cantina)
PRENDI LA CHIAVE	(nessuna difficoltà)
A (o ALTO)	(va nell'atrio)
APRI LA PORTA	(solo se ha la chiave)
E (o EST)	(va nello studio, solo se la porta è aperta)
PRENDI I MICROFILM	(nessuna difficoltà)

Manca un finale, ma ci penseremo dopo. Queste sono dunque le parole che il programma deve per forza conoscere, perché si possa arrivare in fondo all'avventura:

APRI (verbi)

```
PRENDI

CHIAVE          (oggetti)
PORTA
MICROFILM
```

Le parole che il programma conosce sono raccolte in un apposito **dizionario**, che si trova nel programma stesso sotto forma di istruzioni DATA.

```
39970 REM
39980 REM - DIZIONARIO:
39990 REM
40000 DATA "?",13,"A",5,"AGLI",7,"AL",7,"ALL",7
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5
40020 DATA "B",6,"BASSO",6
40030 DATA "COL",7,"CON",7,"COSA",13
40040 DATA "E",3,"EST",3
40050 DATA "GLI",7,"GUARDA",10
40060 DATA "I",7,"IL",7,"INVENTARIO",13
40070 DATA "L",7,"LA",7,"LASCIA",9
40080 DATA "LE",7,"LO",7,"LOAD",12
40090 DATA "N",1,"NORD",1
40100 DATA "O",4,"OVEST",4
40110 DATA "POSA",9,"PRENDI",8
40120 DATA "RIPRENDI",12
40130 DATA "S",2,"SALI",5,"SALVA",11
40140 DATA "SAVE",11,"SCENDI",6,"SUD",2
40150 DATA "UN",7,"UNA",7,"UNO",7
40160 DATA "W",4
40170 DATA "FD":REM --- RICORDARSI L'ORDINE ALFABETICO! ---
```

Notiamo che il vocabolario contiene già:

- Le sei direzioni (difatti il programma le capisce).
- Articoli e preposizioni (che vengono semplicemente ignorati).
- Il verbo GUARDA.
- I verbi PRENDI e LASCIA (o POSA).
- I comandi SALVA (o SAVE) e RIPRENDI (o LOAD).
- Il comando COSA (o INVENTARIO, o '?').

Dunque, PRENDI è già contenuto nel dizionario. Ci restano da inserire APRI, CHIAVE, PORTA e MICROFILM. Per farlo, occorre assegnare un **codice** a ciascuna parola.

Mi spiego: osservando il dizionario, potete vedere che ogni parola è seguita da un numero: il suo codice, appunto. Per esempio, LASCIA (linea 40070) è seguita dal numero 9: vuol dire che 9 è il codice di LASCIA.
In tutto il resto del programma, non verrà mai usata la parola LASCIA, ma il suo codice (9).

Questo permette un notevole risparmio di tempo e memoria, ed inoltre rende semplice introdurre sinonimi, cioè parole con lo stesso significato. Ad esempio, nella linea 40110 vedete che anche la parola POSA ha codice 9. Per dire che due parole sono equivalenti, basta dare il medesimo codice.

Dobbiamo dunque assegnare un **codice** a ciascuna delle parole usate.

Non c'è motivo particolare di usare un numero piuttosto che un altro, ma ci sono queste limitazioni:

- I codici ammessi vanno da 1 a 98.
- I codici da 1 a 13 sono già usati.

Inoltre, io ho trovato comodo (ma non è per niente obbligatorio) raggruppare i vocaboli dello stesso tipo. Ad esempio, in "L'anello di Lucrezia Borgia" ho usato i codici da 20 in poi per i verbi, quelli da 40 in poi per gli oggetti prendibili (come l'anello) e quelli da 60 in poi per gli oggetti non prendibili, o i personaggi (come la strega). Usando lo stesso sistema, darei questi codici:

```
APRI         20
CHIAVE       40
MICROFILM    41
PORTA        60
```

Stabiliti i codici, non resta che introdurre in dizionario le nuove parole.

C'è una regola fondamentale da rispettare: **l'ordine alfabetico**.

Cominciamo con l'inserire `APRI` al suo posto, tra virgolette e seguito dal suo codice (20):

```
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5,"APRI",20
40020 DATA "B",6,"BASSO",6
```

Dato che `APRI` sta (in ordine alfabetico) tra `ALTO` e `B`, va inserito in mezzo tra questi due vocaboli. È la stessa cosa separarlo dal precedente con una virgola o metterlo in una linea tutta per sé:

```
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5
40015 DATA "APRI",20
40020 DATA "B",6,"BASSO",6
```

Quest'ultima soluzione è più comoda, la precedente era più compatta. Dato che l'avventura è molto piccola e mi interessa la chiarezza, userò di preferenza la seconda.

Occorre fare attenzione a non mettere virgole in fondo alle linee: le virgole servono solo per separare i dati tra loro, come se fossero su linee distinte.

Le parole vanno in MAIUSCOLO.

Se avessimo dovuto inserire, che so, `ALLEATO`, sarebbe stato necessario modificare la linea 40010 (non fatelo, è solo un esempio):

```
40010 DATA "ALLA",7,"ALLE",7,"ALLEATO",66,"ALLO",7,"ALTO",5
```

o spezzarla:

```
40010 DATA "ALLA",7,"ALLE",7,"ALLEATO",66
40012 DATA "ALLO",7,"ALTO",5
```

Una volta capito il meccanismo, è molto semplice. Per modificare il programma, assicuratevi di averlo in memoria, oppure fate:

```
LOAD "ZANNA2"
```

ed inserite le nuove parole nel dizionario, che alla fine deve presentarsi in questo modo:

```
39970 REM
39980 REM - DIZIONARIO:
39990 REM
40000 DATA "?",13,"A",5,"AGLI",7,"AL",7,"ALL",7
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5
40015 DATA "APRI",20
40020 DATA "B",6,"BASSO",6
40025 DATA "CHIAVE",40
40030 DATA "COL",7,"CON",7,"COSA",13
40040 DATA "E",3,"EST",3
40050 DATA "GLI",7,"GUARDA",10
40060 DATA "I",7,"IL",7,"INVENTARIO",13
40070 DATA "L",7,"LA",7,"LASCIA",9
40080 DATA "LE",7,"LO",7,"LOAD",12
40085 DATA "MICROFILM",41
40090 DATA "N",1,"NORD",1
40100 DATA "O",4,"OVEST",4
40105 DATA "PORTA",60
40110 DATA "POSA",9,"PRENDI",8
40120 DATA "RIPRENDI",12
40130 DATA "S",2,"SALI",5,"SALVA",11
40140 DATA "SAVE",11,"SCENDI",6,"SUD",2
40150 DATA "UN",7,"UNA",7,"UNO",7
40160 DATA "W",4
40170 DATA "FD":REM --- RICORDARSI L'ORDINE ALFABETICO! ---
```

Ora salvate il programma, con

```
SAVE "ZANNA3"
```

e date il RUN. Dopo il solito inizio, scrivete:

Cosa devo fare ? GRUNT

- Non conosco il verbo 'GRUNT'.

Questo vuol dire: la parola GRUNT (che considero un verbo, o un comando, perché è la prima che incontro) non è nel mio dizionario.

Se la parola sconosciuta è al secondo posto (gli articoli non contano), la risposta è diversa:

Cosa devo fare ? GUARDA IL GRUNT

- Non conosco la parola 'GRUNT'.

Il che, implicitamente, significa che conosce la parola GUARDA (infatti, c'è nel dizionario). Ora scriviamo una frase sensata:

Cosa devo fare ? APRI LA PORTA

- Non capisco.

Perché non capisce? Perché conosce le singole parole APRI e PORTA, ma non l'azione descritta dalle due parole combinate. Dovremo insegnargliela, ma più tardi.

Per adesso, notate che ci sono alcune azioni già predisposte e funzionanti (le ho introdotte io, per facilitarvi il lavoro), ad esempio:

Cosa devo fare ? PRENDI LA CHIAVE

- Qui non ne vedo.

Ha ragione: non c'è nessuna chiave. Abbiamo difatti inserito la parola CHIAVE, ma non l'oggetto corrispondente. Dobbiamo ancora farlo.

Prima di passare ad occuparci degli oggetti, fate un esperimento: inserite un vocabolo fuori dell'ordine alfabetico e fate partire il programma. Guardate cosa succede.

La tavola degli oggetti

Perché CHIAVE non sia una semplice parola, ma un oggetto che fa parte dell'avventura, dobbiamo inserire le caratteristiche di questo oggetto nell'apposita tavola, costituita da un'elenco di linee di DATA.

Per ora, l'elenco è vuoto:

```
54970 REM
54980 REM - OGGETTI:
54990 REM
55000 DATA "FO":REM --- DESCRIZIONI E DATI OGGETTI, DAL N.1
```

È molto simile alla mappa: una serie di linee di DATA, terminate da "FO". Ogni linea ha questo aspetto:

```
55000 DATA "una chiave di sicurezza",40,4
```

C'è la descrizione dell'oggetto (usata per "Vedo..."), e ci sono due numeri.

Il primo numero è il codice (nel dizionario) della parola che descrive l'oggetto.

Infatti 40 è il codice di CHIAVE nel dizionario.
In altri termini: il primo oggetto risponde alla parola CHIAVE. Invece di scrivere CHIAVE nell'elenco oggetti, ne scrivo il codice, cioè 40.

Il secondo numero è il luogo in cui l'oggetto inizia il gioco.

Dato che la chiave si trova inizialmente in cantina, il secondo numero della linea DATA è 4 (ricordo che la cantina è il luogo numero 4).

Non c'è alcun ordine fisso per gli oggetti (alfabetico o altro), ma bisogna ricordarsi che sono identificati dalla loro posizione nell'elenco: il primo è il numero 1, il secondo il numero 2, ecc.

Quindi la chiave è l'oggetto numero 1, in quanto è il primo dell'elenco. Non fate confusione con il 40, che è il codice della parola `CHIAVE`.

Eventuali nuovi oggetti vanno dunque aggiunti in fondo, per non cambiare i numeri di quelli già introdotti. Tanto perché lo sappiate, il programma elenca gli oggetti visibili nell'ordine in cui li trova nei DATA.

Dunque, ecco l'elenco dei nostri oggetti:

```
55000 DATA "una chiave di sicurezza",40,4
55010 DATA "dei microfilm arrotolati",41,3
55020 DATA "una porta blindata",60,2
55030 DATA "FO"
```

Ancora una volta, non dimenticate la chiusura "`FO`". Dato l'ordine in cui sono messi, gli oggetti hanno questi numeri, che serviranno per identificarli in tutto il resto del programma:

1) chiave di sicurezza.
2) microfilm arrotolati.
3) porta blindata.

Introdotto l'elenco degli oggetti, è il caso di salvare il programma:

```
KILL "ZANNA1.BAS"
SAVE "ZANNA4"
```

Il `KILL "ZANNA1.BAS"` serve a fare un po' di spazio su disco, per non rischiare di riempirlo: le ultime tre versioni del programma sono una garanzia sufficiente di sicurezza.

A proposito di sicurezza, è il caso di fare un `SAVE "ZANNA4"` anche sul secondo dischetto che avevate preparato (vero?).

Salvando sul secondo dischetto ogni tre versioni, e conservando il secondo dischetto in un posto diverso dal primo, avete la massima sicurezza di non perdere tutto il vostro lavoro per uno stupido inconveniente.

D'ora in poi, non vi dirò più di salvare il programma periodicamente. Se avete capito l'antifona, bene, se no affari vostri. Tanto, non sarò io a doverlo riscrivere daccapo.

Pronti a provare? Date il `RUN`:

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.
```

La porta c'è. Andiamo a vedere il resto:

```
Cosa devo fare ? B
```

```
Sei in una cantina ammuffita.
```

```
Vedo una chiave di sicurezza.
```

Come dicevo, le azioni `PRENDI` e `LASCIA` sono già predisposte, dunque è come fossero automatiche:

```
Cosa devo fare ? PRENDI LA CHIAVE

Fatto.
```

E possiamo verificare:

```
Cosa devo fare ? COSA
Possiedi:

- una chiave di sicurezza.
```

Tutto bene, dunque? Quasi:

```
Cosa devo fare ? A

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? PRENDI LA PORTA

Fatto.

Sei in un atrio arredato con stile.

Cosa devo fare ? COSA

Possiedi:

- una chiave di sicurezza.
- una porta blindata.
```

Va be' che un agente segreto è sempre in buona forma fisica, ma portarsi sottobraccio una porta blindata mi pare eccessivo.

Come si fa ad impedire che la porta blindata possa essere presa tranquillamente, come qualsiasi altro oggetto? Torniamo all'elenco degli oggetti:

```
55000 DATA "una chiave di sicurezza",40,4
55010 DATA "dei microfilm arrotolati",41,3
55020 DATA "una porta blindata",60,2
55030 DATA "FO"
```

L'ultimo numero di ogni linea, come già visto, indica il luogo iniziale dell'oggetto. Ecco tutti i luoghi possibili:

1..98	il luogo corrispondente, prendibile.
-1..-98	il luogo corrispondente, non prendibile.
-99	il Limbo.
0	trasportato dall'avventuriero.

Quindi, se un oggetto si trova nel luogo zero, significa che è in possesso dell'avventuriero.

Se si trova nel luogo -99, è nel Limbo, cioè in una specie di deposito fuori del gioco, da cui può essere ripescato quando si vuole.

Se infine il luogo di un oggetto è negativo, vuol dire che l'oggetto non può essere preso.

Quest'ultimo è proprio il caso della nostra porta blindata, dunque correggiamo aggiungendo un segno meno:

```
55000 DATA "una chiave di sicurezza",40,4
55010 DATA "dei microfilm arrotolati",41,3
55020 DATA "una porta blindata",60,-2
55030 DATA "FO"
```

e proviamo (RUN):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? PRENDI LA PORTA

- Non e' possibile.
```

Così va meglio. Però:

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? E

Sei nello studio dell'Ambasciatore.
Vedo dei microfilm arrotolati.
```

La porta blindata non blocca il passaggio! Per forza, come fa il programma a saperlo se non glielo spieghiamo?

È ora di passare all'azione, ed a questo è dedicato il prossimo capitolo.

CAPITOLO 8

UNA PORTA CHIUSA A CHIAVE

Il programma deve conoscere le azioni che il giocatore può compiere, ciascuna delle quali risponde ad una certa frase in un certo luogo. L'effetto di un'azione consiste, in genere, nella stampa di un messaggio e nella modifica della posizione di uno o più oggetti.

Torniamo alla nostra porta blindata. Deve aprirsi in risposta alla frase:

`APRI LA PORTA`

ma solo se l'avventuriero possiede la chiave (che abbiamo messo in cantina).

La prima cosa da fare è insegnare al programma a riconoscere la frase `APRI LA PORTA`. Dato che gli articoli sono ignorati, la frase si riduce in realtà a:

`APRI PORTA`

C'è un altro punto da considerare: la frase deve valere soltanto nell'atrio, cioè nel luogo 2, dove si trova la porta (vedremo poi come prevedere azioni valide in tutti i luoghi). Riassumendo:

Luogo:	Prima parola:	Seconda parola:
`ATRIO`	`APRI`	`PORTA`

Ora sostituiamo il luogo col suo <u>numero</u>, e le parole con i rispettivi <u>codici</u>.
L'atrio è il luogo numero 2, `APRI` e `PORTA` hanno rispettivamente codice 20 e 60 nel dizionario:

Luogo:	Prima parola:	Seconda parola:
`2`	`20`	`60`

Portiamo a due cifre tutti i numeri. In questo caso, è solo il numero del luogo che richiede una modifica:

Luogo:	Prima parola:	Seconda parola:
`02`	`20`	`60`

e finalmente uniamo i tre numeri, ottenendo un unico numero di sei cifre:

`022060`

Questo è il **codice della frase** `APRI LA PORTA` nel luogo 2 (l'atrio).

Dato che si tratta di un numero (e non di una stringa), lo zero iniziale può essere tralasciato (insomma, in questo caso era inutile aggiungerlo):

```
22060
```

Conviene, comunque, abituarsi a fare questo lavoro con sei cifre: due per il luogo, due per la prima parola e due per la seconda parola, togliendo alla fine lo zero o gli zeri iniziali di troppo. È meno facile sbagliarsi.

Che ce ne facciamo di questo numero? Scommetto che avete già indovinato: lo introduciamo nell'apposita **tavola delle azioni**. Al momento, ci sono alcune azioni già predisposte:

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

Ogni azione nella tavola è composta da due numeri: il **codice della frase** ed il **numero dell'azione**.

Il primo non è altro che il numero che abbiamo appena preparato, il secondo merita un'ulteriore spiegazione. Per il momento, diciamo che è un numero compreso tra 8 e 160 e che conviene usare il primo numero libero. Dato che non ne abbiamo ancora impiegato nessuno, usiamo 8.

L'azione va introdotta nella tavola rispettando l'**ordine numerico** dei codici di frase (il numero di azione non conta).

Se ci si sbaglia, niente paura: il programma non parte e segnala un errore, indicando i primi due codici fuori ordine (esattamente come succede con il dizionario).

Dunque, introduciamo la nostra azione numero 8 (`APRI LA PORTA`, nell'atrio):

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 DATA 22060,8
50030 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

e facciamo partire il programma (`RUN`):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

8
```

Che vuol dire? Semplice: il programma ha riconosciuto la frase ed ha tentato di eseguire l'azione numero 8 (che dobbiamo ancora scrivere).

Verifichiamo che l'azione funzioni solo nell'atrio:

Cosa devo fare ? N

Sei in un'ampia camera da letto.

Cosa devo fare ? APRI LA PORTA

- Non capisco.

Infatti, in camera da letto non c'è nessuna porta.
Ora che la frase viene riconosciuta, e solo nel luogo voluto, è il momento di occuparsi dell'azione vera e propria.

Passiamo all'azione

Il numero 8 che viene stampato in risposta alla frase `APRI LA PORTA` (pronunciata nell'atrio) viene da qui:

```
6390 REM 8:
6400 PRINT 8:RETURN
6590 REM 9:
6600 PRINT 9:RETURN
6790 REM 10:
6800 PRINT 10:RETURN
....
```

In risposta alla nostra frase, il programma ha eseguito automaticamente un GOSUB 6400.

Se avessimo messo 9 come numero dell'azione, avrebbe eseguito un GOSUB 6600, con 10 un GOSUB 6800, e così via fino all'azione numero 160, che fa eseguire un GOSUB 36800 (vedi il listato del Modulo Base nell'Appendice A).

L'effetto di un'azione consiste nell'eseguire le corrispondenti linee BASIC.

Nel nostro caso, l'azione numero 8 consiste nell'eseguire le linee BASIC che vanno dalla 6400 fino alla prima istruzione RETURN, che termina l'azione.

La linea corrispondente ad una data azione è facilmente identificabile grazie al numero riportato sulla linea precedente linea di REM.

Non è troppo chiaro? Modificate la linea 6400:

```
6400 PRINT "Ehi, funziona!": RETURN

RUN
```

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

Ehi, funziona!

Visto? Possiamo costruire l'azione numero 8 come ci pare e piace. Possiamo ottenere tutto quello che vogliamo. Il problema, naturalmente, è sapere cosa vogliamo.
La prima cosa da fare è verificare se l'avventuriero ha la chiave. In caso contrario, la porta non può essere aperta. Diciamolo in modo più formale:

- Se non ha la chiave, stampa "Non hai la chiave" e termina l'azione.
- Altrimenti, apre la porta e termina l'azione.

Come si fa a sapere se la chiave è in possesso dell'avventuriero? C'è un array che contiene il luogo di ciascun oggetto.

Se non sapete cos'è un array (è roba che si mangia?), niente paura: per usarlo non occorre sapere com'è fatto. Basta sapere che:

`L(1)` è il luogo dell'oggetto 1
`L(2)` è il luogo dell'oggetto 2
`L(3)` è il luogo dell'oggetto 3
(ecc.)

Per capire meglio la cosa, fate partire il programma (`RUN`), poi fermatelo con il solito CTRL-C e scrivete:

```
PRINT L(1)
4
```

Dice che il luogo dell'oggetto 1 è 4, cioè che l'oggetto 1 si trova nel luogo 4.

È vero? Sì, l'oggetto 1 è la chiave (vedi cap.7) e il luogo 4 è la cantina.

Adesso lo freghiamo noi: gli cambiamo il luogo dell'oggetto 1:

```
L(1)=0
```

e lo facciamo ripartire:

```
GOTO 760

Sei in atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? COSA

Possiedi:

- una chiave di sicurezza.
```

Cosa abbiamo fatto? Abbiamo messo l'oggetto 1 (la chiave) nel luogo zero.
Ma il luogo zero è speciale: vuol dire "trasportato dall'avventuriero", quindi abbiamo dato la chiave in mano all'agente segreto!

Con questo sistema, possiamo manipolare tutti gli oggetti del gioco a nostro piacimento (come dice la nonna: Incredibile! con queste diavolerie moderne, chissà dove andremo a finire).

Riassumendo, per sapere se l'avventuriero ha la chiave (in altri termini: se la chiave è trasportata), basta l'istruzione BASIC:

```
IF L(1)=0 THEN...
```

Cioè: se (IF) l'oggetto 1 (la chiave) si trova nel luogo zero (trasportata), allora (THEN) esegue il resto della linea, altrimenti lo salta.

Tornando alla nostra azione 8 in costruzione:

```
6390 REM 8:
6400 IF L(1)<>0 THEN PRINT "Non hai la chiave.":RETURN
6410 PRINT "Ok":RETURN
```

Dunque: se non ha la chiave (il luogo dell'oggetto 1 è diverso da zero, dunque non è trasportato), stampa "`Non hai la chiave`" e termina l'azione con RETURN.

Un'istruzione RETURN conclude l'azione immediatamente.

In caso contrario (quindi, se ha la chiave), prosegue alla linea successiva, che stampa "`Ok`" e termina l'azione. Proviamolo (`RUN`):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

Non hai la chiave.

Cosa devo fare ? B

Sei in una cantina ammuffita.
Vedo una chiave di sicurezza.
Cosa devo fare ? PRENDI LA CHIAVE

Fatto.

Cosa devo fare ? A

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

Ok
```

Funziona. Per adesso si limita a stampare "`Ok`", ma funziona: apre la porta solo se ha la chiave.

Adesso bisogna far aprire effettivamente la porta. Come si può fare?

Sarebbe una buona cosa modificare "una porta blindata" in "una porta blindata aperta", ma il Modulo Base non consente di cambiare la descrizione di un oggetto (per gli esperi: si può fare, ma occorre ricordarsi di registrarla). E allora?

Allora, c'è un trucco molto semplice: aggiungiamo un nuovo oggetto all'elenco:

```
55000 DATA "una chiave di sicurezza",40,4
55010 DATA "dei microfilm arrotolati",41,3
55020 DATA "una porta blindata",60,-2
55030 DATA "una porta blindata aperta",60,-99
55040 DATA "FO"
```

Ricordo che i nuovi oggetti vanno aggiunti in fondo all'elenco, per non alterare il numero dei precedenti.

Abbiamo aggiunto l'oggetto numero 4, con descrizione "una porta blindata aperta", che risponde alla parola PORTA (codice 60 nel dizionario) e che si trova nel luogo -99.

Se l'avete dimenticato, il luogo -99 è il Limbo, cioè il "deposito" degli oggetti fuori gioco. All'inizio dell'avventura, quindi, l'oggetto 4 non c'è.

Dato che siete svegli, avrete già capito il trucco: per aprire la porta, scambiamo due oggetti: facciamo sparire l'oggetto 3 (porta chiusa) mettendolo nel Limbo (-99) e facciamo apparire l'oggetto 4 (porta aperta) mettendolo nell'atrio (luogo 2).
Si fa in un attimo:

```
6390 REM 8:
6400 IF L(1)<>0 THEN PRINT "Non hai la chiave.":RETURN
6410 PRINT "Ok":L(3)=-99:L(4)=-2:RETURN
```

(Notate il segno meno davanti al 2, perché la porta non è prendibile). Date il solito RUN, andate a prendervi la chiave (B, PRENDI LA CHIAVE, A) e provate:

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

Ok

Sei in un atrio arredato con stile.
Vedo una porta blindata aperta.

Noi sappiamo che sono stati scambiati due oggetti (il 3 ed il 4), ma il giocatore non se ne accorge: per lui è sempre il medesimo oggetto (la porta), che ha cambiato stato (da chiusa ad aperta).

E la porta funziona. Se vi sembra che ci sia voluto tanto lavoro, è solo perché ho descritto tutto nei minimi particolari: in realtà, sono soltanto quattro righe di programma (6400, 6410, 50020 e 55030).

Resta una cosa da fare: la porta non blocca il passaggio! Bisogna fare in modo che lo spostamento dall'atrio allo studio sia permesso solamente se la porta è aperta.

Quando interviene la porta? Quando il giocatore fa EST (o E) dall'atrio. Dobbiamo quindi scrivere un'azione che risponde alla frase EST pronunciata nell'atrio.

Ma questa non va ad interferire con il regolare funzionamento della direzione EST? No, perché la frase EST pronunciata in un certo luogo è diversa dalla frase EST generica (valida in tutti i luoghi), ed ha la precedenza su questa.

Possiamo dunque costruire il codice di azione:

Luogo:	Prima parola:	Seconda parola:
ATRIO	EST	(nessuna)
02	03	00

Due note: il codice di EST è 3, come si vede dal vocabolario (linea 40040), e lo portiamo a due cifre (03). Se la frase dev'essere composta di una sola parola, si indica 00 come seconda parola. Il codice dell'azione EST nell'atrio è dunque:

020300

cioè 20300 (gli zeri a sinistra si possono togliere).

Mettiamolo nella tavola delle azioni, indicando il numero dell'azione da eseguire in risposta alla frase EST nell'atrio (prendiamo la prima libera: la numero 9):

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 DATA 20300,9,22060,8
50030 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

Era la stessa cosa scrivere una nuova linea di DATA, invece di aggiungere l'azione in una linea preesistente. L'importante è rispettare l'ordine numerico.

Adesso sappiamo (se avete dubbi, provatelo) che la frase EST, scritta nell'atrio, causa l'esecuzione della routine di azione numero 9 (linea 6600).

Cosa deve fare l'azione numero 9? Semplice:

- Se la porta è chiusa, stampa "La porta è chiusa" e termina l'azione.
- Se la porta è aperta, lascia passare nello studio e termina l'azione.

Per sapere se la porta è chiusa, basta guardare il luogo dell'oggetto 3 (la porta chiusa): se l'oggetto 3 è nel luogo -2 (l'atrio, '-' perché non prendibile), la porta è ancora chiusa, se è nel luogo -99 la porta è stata aperta.

Si potrebbe fare la stessa cosa, a rovescio, usando l'oggetto 4 (la porta aperta). Infatti, come ricordate, l'apertura della porta consiste nel far sparire l'oggetto 3 (la porta chiusa) ed apparire l'oggetto 4 (porta aperta). Noi usiamo il primo sistema:

```
6590 REM 9:
6600 IF L(3)=-2 THEN PRINT "La porta e' chiusa.":RETURN
6610 ???
```

(ricordate il segno meno in -2). Ma cosa vuol dire "lascia passare nello studio? Vuol dire spostare l'avventuriero nello studio, cioè nel luogo 3. Ma questo, se

ricordate, lo sappiamo già fare: basta mettere il numero del nuovo luogo nella variabile LU (vedi Cap. 6). Quindi:

```
6590 REM 9:
6600 IF L(3)=-2 THEN PRINT "La porta e' chiusa.":RETURN
6610 LU=3:RETURN
```

Proviamo: fate RUN ed andate a prendervi la chiave, poi tornate nell'atrio:

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? E

La porta e' chiusa.

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA

Ok

Sei in un atrio arredato con stile.
Vedo una porta blindata aperta.

Cosa devo fare ? E

Sei nello studio dell'Ambasciatore.
Vedo dei microfilm arrotolati.
```

Finalmente ci siamo. La sequenza principale è a posto e funziona. Adesso non resta che arricchirla con un po' di dettagli.

Nel prossimo capitolo approfondirò anche le possibilità offerte dalle varie combinazioni possibili per il codice di azione.

Una sintesi di tutto questo si trova, come al solito, nell'Appendice B. Una volta finito il libro, potrete usare l'Appendice B come guida rapida di riferimento, senza dover sfogliare molte pagine alla ricerca delle informazioni che vi servono.

CAPITOLO 9

I DETTAGLI CONTANO

La qualità di un'avventura dipende molto dal livello di dettaglio. Un buon autore dedica parecchio tempo a rifinire ed aggiungere varianti, aiutandosi con i suggerimenti forniti dai collaudatori a cui fa provare il gioco.

Come dicevo, la sequenza principale è a posto e funziona. Ma l'avventura che ne risulta non è certo interessante: il giocatore deve sorbirsi un'infinità di "non capisco", e le uniche azioni valide sono quelle che portano direttamente alla soluzione dell'avventura. Purtroppo ci sono in circolazione troppe avventure che non si discostano molto da questo sistema: o si fa l'azione giusta, o niente.

Contrariamente a quanto sembrano pensare alcuni autori, perché un'avventura sia divertente non serve che abbia una mappa particolarmente estesa, o che presenti problemi difficilissimi. Deve invece essere molto curata nei dettagli, perché sono proprio questi a dare al giocatore l'illusione di vivere realmente dentro l'avventura, in un universo fantastico che voi avete creato.

C'è però un prezzo da pagare: la rifinitura dei dettagli è la parte più laboriosa dell'intero lavoro, e potete aspettarvi di impiegare per questa oltre la metà del tempo complessivo. Per essere sicuri di ottenere un buon risultato, ci sono due cose da fare:

- Aggiungere tutti i dettagli che vi vengono in mente.
- Far collaudare l'avventura, ed aggiungere nuovi dettagli.

Iniziamo dalla prima: cosa manca nella nostra micro-avventura? Per prima cosa, il finale. È ovvio che ci vuole una qualche forma di conclusione.

Gran finale

Non cadete nell'altro errore tipico degli sceneggiatori trascurati: il finale insulso del tipo "congratulations, you win." (bravo, hai vinto).
Dopo che uno ha speso sangue sudore e lacrime per risolvere l'avventura, ha diritto ad aspettarsi qualcosa di più gratificante. Direi che merita almeno una decina di righe (meglio se di più) dedicate all'apoteosi della sua impresa.
Dato però che non ho intenzione di scrivere una vera e propria avventura, mi limiterò a darvi uno spunto, che completerete voi se vorrete espandere l'esempio in un'avventura vera e propria (un ottimo e divertente esercizio).

Ma prima di tutto, dove mettiamo il finale? Quando l'agente trova i microfilm? No, troppo presto: potrebbe ancora essere colto in flagrante. Bisogna uscire sani e salvi dall'appartamento, e solo allora la missione può dirsi compiuta.

Come facciamo ad impedire che l'avventuriero esca dall'appartamento prima di aver trovato i microfilm? Non c'è alcun motivo per impedirglielo, soltanto la sua missione fallisce miseramente se sceglie di farlo. E, si sa, un agente segreto che non sia in grado di compiere una missione è del tutto inutile, anzi è un intralcio di cui liberarsi al più presto...

Dunque, per concludere la missione occorre uscire dall'appartamento.

L'uscita è ad ovest dell'atrio, e non occorre che la mettiamo nella mappa, perché basta introdurre l'azione "Ovest" nel luogo 2 (l'atrio, appunto), così:

Luogo:	Prima parola:	Seconda parola:
ATRIO	OVEST	(nessuna)
02	04	00

Da cui:

020400, cioè 20400

che facciamo corrispondere alla prima azione libera, la numero 10:

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 DATA 20300,9,20400,10,22060,8
50030 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

Ora si tratta di scrivere l'azione numero 10.

Se l'agente ha trovato i microfilm, la missione è riuscita, altrimenti è miseramente fallita.
Per evitare di uscire per errore, diamo al giocatore la possibilità di confermare od annullare la decisione di uscire:

```
6790 REM 10:
6800 A$="Vuoi lasciare l'appartamento":GOSUB 1600
6810 IF NOT (A) THEN RETURN
6820 PRINT:IF L(2)=0 GOTO 6850:REM Ok
6830 PRINT "Fedeli agli ordini, i tuoi colleghi"
6840 PRINT "eliminano un agente incapace.":GOTO 4000
6850 GOSUB 1500:REM <spazio>
6860 PRINT:PRINT "- Missione compiuta! -"
6870 PRINT "Hai salvato il tuo Paese e ti"
6880 PRINT "sei guadagnata la promozione a:":PRINT
6890 PRINT "         ######":PRINT
6900 PRINT "(il tuo nuovo titolo e' cosi' segreto"
6910 PRINT "che non posso nemmeno nominarlo).":PRINT
6920 PRINT "Congratulazioni, ######!":PRINT:END
```

C'è parecchio da dire. Iniziamo dalle linee 6800-6810. La 6800 chiama la subroutine 1600, che si occupa di porre una domanda, e torna solo quando il giocatore ha risposto SI o NO (o almeno l'iniziale S o N).

La domanda da porre va messa nella variabile A$ (senza spazi o punto di domanda).

Al ritorno dalla subroutine, la variabile logica A conterrà il valore vero se la risposta era SI, falso se era NO.

La linea 6810 significa dunque: se la risposta era NO (NOT vero è evidentemente falso), ritorna senza fare nulla.

Se possiede i microfilm, cioè se il luogo dell'oggetto 2 (i microfilm) è zero (trasportati), la linea 6820 salta alla 6850, dove viene stampato il finale positivo (missione compiuta), preceduto da un "premi <spazio> per continuare" (che si ottiene con un GOSUB 1500). Alla fine, un END conclude il programma.

Se invece non ha i microfilm, il programma prosegue diritto alla 6830, dove stampa un messaggio di fallimento, e poi salta alla 4000.

La routine che inizia alla linea 4000 è quella del finale negativo (corrispondente di solito, anche se non sempre, alla morte dell'avventuriero).

Un GOTO 4000 conclude l'avventura quando è finita male.

Al momento, la routine fa ben poco:

```
3970 REM
3980 REM - MORTO -
3990 REM
4000 GOSUB 1500:REM <SP>
4010 REM --- METTERE QUI IL FINALE NEGATIVO ---
4450 A$="Vuoi giocare ancora":GOSUB 1600:IF(A) THEN RUN:REM
RIPARTE
4460 PRINT:PRINT:PRINT "Ciao!":PRINT:END
```

Si possono introdurre in questa routine delle linee di PRINT che stampino un messaggio, alla fine del quale il programma chiede se ripartire dall'inizio o smettere di giocare (linea 4450).

Notate notate l'uso della subroutine 1600 per porre una domanda.

Anche qui ci sta bene un'adeguata conclusione, di cui mi limito a fornire un breve spunto:

```
3970 REM
3980 REM - MORTO -
3990 REM
4000 GOSUB 1500:REM <SP>
4010 PRINT:PRINT "La tua missione e' fallita."
4020 PRINT "Di te non restera' che un"
4030 PRINT "polveroso fascicolo, presto"
4040 PRINT "dimenticato.":PRINT
4450 A$="Vuoi giocare ancora":GOSUB 1600:IF(A) THEN RUN:REM
RIPARTE
4460 PRINT:PRINT:PRINT "Ciao!":PRINT:END
```

Ora, collaudiamo il tutto (RUN):

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? O

Vuoi lasciare l'appartamento ? NO

Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? O

Vuoi lasciare l'appartamento ? SI

**Fedeli agli ordini, i tuoi colleghi
eliminano un agente incapace.**

- Premi <spazio> per continuare -

**La tua missione e' fallita.
Di te non restera' che un
polveroso fascicolo, presto
dimenticato.**

Vuoi giocare ancora ?

Ora, riprovate il tutto dopo aver preso i microfilm. Se non avete voglia di fare il giro per prendere i microfilm, usate il solito trucco:

- fermate il programma con CTRL-C.

- prendete i microfilm con `L(2)=0`.

- fate ripartire il programma con `GOTO 760`.

Con i microfilm in mano, il finale e' diverso:

**Sei in un atrio arredato con stile.
Vedo una porta blindata.**

Cosa devo fare ? O

Vuoi lasciare l'appartamento ? SI

- Premi <spazio> per continuare -

**- Missione compiuta! -
Hai salvato il tuo Paese e ti
sei guadagnata la promozione a:**

#####

**(il tuo nuovo titolo e' cosi' segreto
che non posso nemmeno nominarlo).**

Congratulazioni, #####!

Ed il finale è a posto.
Noterete, scrivendo avventure, che una delle cose più noiose è la necessità di controllare l'effetto dei PRINT sul video: bisogna, ogni volta, far ripartire il programma, procurarsi gli oggetti, andare nel luogo giusto, ecc.

C'è un sistema più semplice: chiamare direttamente le linee che eseguono la stampa.

Per esempio, volendo controllare il finale negativo basta fare un GOSUB (a programma fermo) alla prima linea di PRINT:

```
GOSUB 4010
```

La tua missione e' fallita.
Di te non restera' che un
polveroso fascicolo, presto
dimenticato.

Questa tecnica è comodissima, anche se a volte può dare messaggi di errore del tipo ?BAD SUBSCRIPT ERROR se non avete prima fatto girare il programma. In ogni caso, non c'è nessuna conseguenza negativa.

Guardarsi bene in giro

Una delle azioni fondamentali in ogni avventura è GUARDA. È lecito aspettarsi che un giocatore guardi ogni oggetto, alla ricerca di informazioni o indizi.

Sarebbe ingiusto e fastidioso rispondere sempre "non noto nulla di particolare".

È invece il caso di prevedere un'azione GUARDA per tutti, o quasi tutti, gli oggetti. Per esempio, possiamo usare la prima azione libera (la numero 11) per GUARDA LA CHIAVE:

Luogo:	Prima parola:	Seconda parola:
(dovunque)	GUARDA	CHIAVE

È ovvio che non possiamo limitare la frase GUARDA LA CHIAVE ad un solo luogo, dato che la chiave può essere presa e portata in giro per l'appartamento.

> Per indicare che una frase è valida in tutti i luoghi, si usa zero come numero di luogo.

Dunque, costruiamo il codice dell'azione GUARDA LA CHIAVE:

Luogo:	Prima parola:	Seconda parola:
00	10	40

Cioè 1040. Aggiorniamo la tavola delle azioni:

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1040,11,1099,-4,1100,5,1200,6,1300,7
50020 DATA 20300,9,20400,10,22060,8
50030 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

E mettiamo un messaggio come azione 11:

```
6990 REM 11:
7000 PRINT "E' una Yale di sicurezza.":RETURN
```

Ora, proviamo (RUN), andando per prima cosa in cantina a prendere la chiave e poi scrivendo:

Cosa devo fare ? GUARDA LA CHIAVE

E' una Yale di sicurezza.

Perfetto. O no? Riproviamo senza la chiave:

Cosa devo fare ? GUARDA LA CHIAVE

E' una Yale di sicurezza.

Ahi! Non si accorge che non abbiamo la chiave. Bisogna modificare l'azione 11 per verificare che la chiave sia trasportata.

Dato però che questo è un caso molto comune, ho previsto un sistema automatico per fare il controllo:

Se il numero di azione è negativo, il Modulo Base controlla che la seconda parola della frase rappresenti un oggetto visibile, cioè presente nel luogo o trasportato dall'avventuriero.

Insomma, basta un segno meno davanti al numero di azione:

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1040,-11,1099,-4,1100,5,1200,6,1300,7
50020 DATA 20300,9,20400,10,22060,8
50030 DATA"FA": REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

Riprovando (RUN) senza la chiave:

Cosa devo fare ? GUARDA LA CHIAVE

- Qui non ne vedo.

mentre, con la chiave presa o visibile, funziona tutto come prima.

Perché questo controllo non viene fatto automaticamente su tutte le frasi? Per consentire la massima libertà in frasi come CERCA LA CHIAVE, a cui si può rispondere "cercatela da te" anche se non è presente, tanto per fare un esempio.

Lascio a voi il compito di aggiungere l'azione GUARDA per tutti gli altri oggetti, ricordandovi di mettere un meno davanti al numero di azione quando si tratta di un oggetto trasportabile.

Altre azioni da aggiungere? Quando avete finito le idee, prendete qualche vostro amico, nominatelo ufficialmente "collaudatore di primo livello" e fatelo giocare, prendendo note senza dire nulla: in mezz'ora avrete un notes pieno di frasi a cui non avevate minimamente pensato.

Armatevi di pazienza, ed introducetele (ampliando il vocabolario con le nuove parole). È così che si costruisce una bella avventura.

Azioni per tutti i gusti

Avrete notato che ci sono varie possibili combinazioni per il codice di azione. Ve le descrivo una per una, ricordandovi che potrete poi consultare l'Appendice B per ricordarvele quando vi servono.

Con frasi di una sola parola, sapete già che basta indicare 00 come codice della seconda parola. ad esempio, per `NORD` nel luogo 3:

Luogo:	Prima parola:	Seconda parola:
`03`	`01`	`00`

quindi il codice è 30100.

Ci sono due possibili combinazioni, che elenco in ordine di precedenza:

1 - Parola valida in un certo luogo.
2 - Parola valida ovunque.

La precedenza significa che il programma guarda per prima cosa se è stata prevista l'azione come valida nel luogo corrente dell'avventuriero (come il nostro `EST` nell'atrio).

Se non la trova, guarda se l'azione è stata prevista come valida in tutti i luoghi (come un `EST` generico).

Se non la trova ancora, stampa "non capisco".

Con le frasi di due parole, ci sono quattro combinazioni, che sono esaminate in quest'ordine:

1 - Prima parola + seconda parola, valide in un certo luogo.
2 - Prima parola + parola qualunque, valide in un certo luogo.
3 - Prima parola + seconda parola, valide ovunque.
4 - Prima parola + parola qualunque, valide ovunque.

Il caso 1 è ben noto (`APRI LA PORTA`, nell'atrio), come pure il caso 3 (`GUARDA LA CHIAVE`, dovunque).

Per rendere l'azione valida ovunque, basta mettere zero come numero di luogo.

I casi 2 e 4 (che differiscono tra loro solo per il numero di luogo, che è zero nel caso 4) sono un po' particolari: l'azione viene eseguita comunque, purché ci sia una seconda parola valida, cioè esistente nel dizionario.

Un esempio tipico (che avete usato finora senza saperlo) è `PRENDI`, che viene eseguita qualunque sia la seconda parola che segue.

È utile sapere che in questo caso (frase di due parole) la variabile **`OG`** contiene il numero (posizione nell'elenco) dell'oggetto nominato, o zero se la seconda parola non si riferisce ad un oggetto valido e visibile (presente o trasportato).

È istruttivo esaminare come funzionano le azioni di `PRENDI` (linea 5200) e `LASCIA` (linea 5400), notando che `LASCIA` può essere eseguita anche su

oggetti non presenti (per la massima libertà di risposta), mentre `PRENDI` ha numero di azione negativo e viene quindi chiamata solo se la seconda parola indica un oggetto visibile (presente o trasportato).

Sempre nell'appendice B, trovate le principali variabili usate dal programma, che possono esservi utili nelle vostre routine di azione. Ad esempio, `P1$` e `P2$` contengono rispettivamente la prima e la seconda parola, mentre `C1` e `C2` contangono i rispettivi codici.

Se avete le idee un po' confuse, non perdetevi d'animo. Cominciate a scrivere una piccola avventura, e vedrete che molte cose si chiariranno man mano.

E mi raccomando: fate più errori possibile. Sbagliando s'impara!

CAPITOLO 10

UNA BOMBA AD OROLOGERIA

Il Modulo Base permette anche di inserire nell'avventura il fattore tempo, facendo accadere un dato evento in un momento prefissato od in rapporto ad una o più azioni compiute dall'avventuriero.

Un'avventura può diventare più interessante se si tiene conto del passaggio del tempo. Non il tempo che il giocatore passa a pensare quale azione compiere, perché non è il caso di aggiungere l'angoscia di una risposta veloce ai molti problemi che già il poveretto si trova a dover affrontare. Parlo invece del tempo inteso come numero di mosse compiute: un esempio ormai abusato è quello della lampada la cui batteria si scarica dopo un certo numero di mosse, lasciando l'avventuriero al buio.

Nella nostra avventura, mi pare logico porre un tempo limite come avverrebbe in condizioni reali. Non è il caso di renderlo molto stringente: diciamo 25 mosse, al termine delle quali l'ambasciatore torna a casa.

Il Modulo Base è già predisposto per tener conto del passaggio del tempo. Ci sono cinque timer usabili a piacimento (T1, T2, T3, T4, T5), che funzionano in questo modo:

- Un timer che sta a zero è inattivo, cioè non ha effetto.
- Un timer contenente un numero positivo è in funzione.

Se un timer è in funzione, ad ogni mossa scende di uno e chiama una certa routine (sempre ad ogni mossa), finché non arriva a zero.
Per la precisione:

- Il Timer 1 esegue un GOSUB 3000 ad ogni mossa in cui è attivo.
- Il Timer 2 esegue un GOSUB 3200 ad ogni mossa in cui è attivo.
- Il Timer 3 esegue un GOSUB 3400 ad ogni mossa in cui è attivo.
- Il Timer 4 esegue un GOSUB 3600 ad ogni mossa in cui è attivo.
- Il Timer 5 esegue un GOSUB 3800 ad ogni mossa in cui è attivo.

Dato che i timer contano alla rovescia, per farli partire occorre metterci un numero positivo. Per limitare il gioco a 25 mosse, possiamo fare così:

```
4950 LU=2:T1=25:PRINT:RETURN
```

La linea 4950, come potete controllare sul vostro listato, è l'ultima dell'introduzione.

Abbiamo dunque fatto partire il Timer 1 (T1), che funzionerà per 25 mosse prima di arrivare a zero e fermarsi. Provate (RUN):

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.
```

T1

È apparso un "T1". Se provate a far passare un po' di tempo, ad esempio spostandovi tra una stanza e l'altra, vedrete che dopo 25 mosse il "T1" scomparirà: a forza di togliere 1, il timer è arrivato a zero e si è fermato.

La stampa del "T1" è stata provocata da questa routine:

```
2990 REM T1:
3000 PRINT:PRINT "T1":RETURN
```

Come al solito, possiamo modificare a piacimento la routine. Nel nostro caso dobbiamo semplicemente ignorare il timer (non far niente) se non è arrivato a zero, mentre in quest'ultimo caso accade l'inevitabile:

```
2990 REM T1:
3000 IF T1>0 THEN RETURN
3010 PRINT:PRINT "L'ambasciatore ritorna a casa."
3020 PRINT "E' la fine!":GOTO 4000
```

Dando il `RUN`, noterete che non appare più il "`T1`" (per forza, abbiamo tolto il PRINT).

Dobbiamo ora verificare che dopo 25 mosse torni l'ambasciatore. Per non aspettare 25 mosse, conviene usare il solito trucco:

- Fermare il programma con CTRL-C.
- Abbreviare il tempo rimasto con `T1=4`.

- Ripartire con `GOTO 760`.

Ora il timer arriverà a zero in sole 3 mosse (la quarta è il `GOTO 760`), e stamperà:

```
L'ambasciatore ritorna a casa.
E' la fine!

- Premi <spazio> per continuare -
```

seguito dal solito finale negativo (ottenuto con il GOTO 4000).

Nessuno proibisce di arricchire la vicenda con eventi collegati a certi valori del timer, ad esempio con una sequenza del tipo:

```
3000 IF T1=5 THEN PRINT "Sento un'auto che arriva.":RETURN
3010 IF T1=3 THEN PRINT "Sento aprire una portiera.":RETURN
3020 IF T1=2 THEN PRINT "Sento dei passi.":RETURN
```

seguita dalle istruzioni viste in precedenza.

La presenza di timer attivi (cioè diversi da zero) rallenta un poco il programma, ma vivacizza il gioco. Come al solito, non bisogna abusarne.

Il reattore de "L'astronave condannata" funziona proprio in questo modo, cioè con un timer che viene avviato all'inizio del gioco.

Un esempio più complesso

Vediamo un altro esempio di uso dei timer. Complichiamo la vicenda, mettendo la chiave in una cassaforte che si può aprire soltanto con una bomba ad orologeria mascherata da sveglia.

Abbiamo bisogno di questi oggetti:

- Una sveglia di buona marca.
- Una solida cassaforte.
- Una cassaforte sventrata.
- Un cumulo di macerie.

e delle parole (a cui assegno già i codici di dizionario):

```
- CARICA             21
- SVEGLIA            42
- CASSAFORTE         43
- CUMULO, MACERIE    61
```

Modifichiamo dunque il dizionario:

```
39970 REM
39980 REM - DIZIONARIO:
39990 REM
40000 DATA "?",13,"A",5,"AGLI",7,"AL",7,"ALL",7
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5
40015 DATA "APRI",20
40020 DATA "B",6,"BASSO",6
40025 DATA "CARICA",21,"CASSAFORTE",43,"CHIAVE",40
40030 DATA "COL",7,"CON",7,"COSA",13,"CUMULO",61
40040 DATA "E",3,"EST",3
40050 DATA "GLI",7,"GUARDA",10
40060 DATA "I",7,"IL",7,"INVENTARIO",13
40070 DATA "L",7,"LA",7,"LASCIA",9
40080 DATA "LE",7,"LO",7,"LOAD",12
40085 DATA "MACERIE",61,"MICROFILM",41
40090 DATA "N",1,"NORD",1
40100 DATA "O",4,"OVEST",4
40105 DATA "PORTA",60
40110 DATA "POSA",9,"PRENDI",8
40120 DATA "RIPRENDI",12
40130 DATA "S",2,"SALI",5,"SALVA",11
40140 DATA "SAVE",11,"SCENDI",6,"SUD",2,"SVEGLIA",42
40150 DATA "UN",7,"UNA",7,"UNO",7
40160 DATA "W",4
40170 DATA "FD":REM --- RICORDARSI L'ORDINE ALFABETICO!!! -
```

e la tavola degli oggetti:

```
55000 DATA "una chiave di sicurezza",40,-99
55010 DATA "dei microfilm arrotolati",41,3
55020 DATA "una porta blindata",60,-2
55030 DATA "una porta blindata aperta",60,-99
55040 DATA "una sveglia di buona marca",42,1
55050 DATA "una solida cassaforte",43,-4
55060 DATA "una cassaforte sventrata",43,-99
55070 DATA "un cumulo di macerie",61,-99
55080 DATA "FO"
```

La sveglia si trova, all'inizio, in camera da letto (luogo 1), e la cassaforte sta in cantina (luogo 4, ma col segno meno perché non è prendibile).
La cassaforte sventrata ed il cumulo di macerie sono, ovviamente, nel Limbo (luogo -99).

Notate la modifica alla linea 55000: la chiave non è più in cantina, ma nel Limbo; apparirà solo una volta distrutta la cassaforte.

Provate (RUN), e verificate che la sveglia sia in camera da letto e che si possa prendere, e che la cassaforte sia in cantina, e non prendibile.

Si tratta ora di usare un timer per la sveglia/bomba. Dato che il timer 1 è già impegnato, useremo il timer 2.

L'azione CARICA LA SVEGLIA farà partire il timer con un tempo abbastanza breve, diciamo 5 mosse.

L'azione GUARDA LA SVEGLIA ci dirà se è carica o meno.

Entrambe le azioni sono valide dovunque, richiedono che l'oggetto (la sveglia) sia presente (numero di azione negativo), ed hanno rispettivamente codice 00-21-42 (dovunque, CARICA, SVEGLIA), cioè 2142, e 00-10-42 (dovunque, GUARDA, SVEGLIA), cioè 1042.

Le prime azioni libere sono la 12 e la 13, dunque:

```
49970 REM
49980 REM - AZIONI:
49990 REM
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1040,-11,1042,-13,1099,-4
50015 DATA 1100,5,1200,6,1300,7,2142,-12
50020 DATA 20300,9,20400,10,22060,8
50030 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
```

Scriviamo l'azione 12 (CARICA LA SVEGLIA):

```
7190 REM 12:
7200 IF T2>0 THEN PRINT "Gia' fatto.":RETURN
7210 PRINT "Cric, cric, cric...":T2=5:RETURN
```

e la 13 (GUARDA LA SVEGLIA):

```
7390 REM 13:
7400 IF T2=0 THEN PRINT "E' scarica.":RETURN
7410 PRINT "Ticchetta.":RETURN
```

Nell'azione 12, è necessario controllare che la sveglia non sia già carica, altrimenti il timer viene rimesso a 5 ogni volta (e non è realistico, anche se non pregiudica il gioco).

Ora fate RUN, andate in camera da letto, e provate:

```
Sei in un'ampia camera da letto.
Vedo una sveglia di buona marca.

Cosa devo fare ? GUARDA LA SVEGLIA
```

E' scarica.

Sei in un'ampia camera da letto.
Vedo una sveglia di buona marca.

Cosa devo fare ? PRENDI LA SVEGLIA

Fatto.

Sei in un'ampia camera da letto.

Cosa devo fare ? CARICA LA SVEGLIA

Cric, cric, cric...

Sei in un'ampia camera da letto.

T2

Cosa devo fare ? CARICA LA SVEGLIA

Gia' fatto.

T2

Cosa devo fare ? GUARDA LA SVEGLIA

Ticchetta.

T2

Ed il "T2" continua ad apparire per altri due turni, dopo di che scompare: il timer 2 è arrivato a zero.

Adesso occupiamoci di questo timer:

- Ad ogni turno, scrive un "tic, tac" se ha la sveglia in mano (luogo zero).

- A zero, la bomba esplode, e ci sono tre casi:

  - È nello stesso luogo dell'avventuriero: ovvia conseguenza.
  - È in cantina: distrugge la cassaforte ed appaiono macerie e chiave.
  - È in un altro luogo: appaiono solo le macerie.

Notate che per sapere se la sveglia (oggetto numero 5) è nello stesso luogo dell'avventuriero, occorre tener conto di due casi: trasportata (luogo della sveglia=0) oppure (OR) presente (luogo della sveglia=LU, cioè stesso luogo dell'avventuriero).

Al lavoro sul timer 2:

```
3190 REM T2:
3200 IF L(5)=0 THEN PRINT:PRINT "Tic, tac, tic, tac..."
3210 IF T2>0 THEN RETURN
3220 PRINT:PRINT "[[[[[ BOOM! ]]]]]"
3230 IF L(5)=0 OR L(5)=LU GOTO 4000
3240 IF L(5)=4 THEN L(6)=-99:L(7)=-4:L(1)=4
```

```
3250 L(8)=-L(5):L(5)=-99:RETURN
```

La linea 3200 stampa "Tic, tac..." solo se l'avventuriero sta trasportando la sveglia (altrimenti è troppo lontano per sentirlo).
La 3210 ritorna senza conseguenze se il timer non è a zero.
La 3220 fa scoppiare la bomba quando il timer arriva a zero (senza controllo di luogo, dato che si sente di sicuro in tutto l'appartamento).

Se la sveglia (oggetto 5) è trasportata (luogo 0) o presente (luogo LU), l'agente salta in aria (3230, si potrebbe aggiungere un messaggio), altrimenti prosegue.

Se la sveglia (oggetto 5) si trova in cantina (luogo 4), scompare la cassaforte (oggetto 6) ed appaiono la cassaforte sventrata (oggetto 7, segno meno perché non prendibile) e la chiave (oggetto 1).

Si potrebbe anche non far apparire la chiave, e richiedere che il giocatore scriva `GUARDA LA CASSAFORTE` o, meglio ancora, `GUARDA LE MACERIE`.

La 3250 fa, in ogni caso, apparire le macerie (oggetto 8) e sparire la sveglia (oggetto 5).
Ancora una volta, attenti al segno meno: dato che le macerie non sono prendibili, devono apparire nel luogo `-L(5)`, non nel luogo `L(5)`.

Ho usato `L(5)` invece di un numero fisso, perché non è possibile sapere dove esplode la bomba, se non guardando dove si trova la sveglia.

È ora di provare: fate `RUN,` andate a prendere la sveglia, e scendete in cantina:

Sei in una cantina ammuffita.
Vedo una solida cassaforte.

Cosa devo fare ? CARICA LA SVEGLIA

Cric, cric, cric...

Sei in una cantina ammuffita.
Vedo una solida cassaforte.

Tic, tac, tic, tac...

Cosa devo fare ? LASCIA LA SVEGLIA

Fatto.

Sei in una cantina ammuffita.
Vedo una sveglia di buona marca.
Vedo una solida cassaforte.

Cosa devo fare ? GUARDA LA SVEGLIA

Ticchetta. (meglio andarsene, e di corsa)

Sei in una cantina ammuffita.
Vedo una sveglia di buona marca.
Vedo una solida cassaforte.

Cosa devo fare ? A

```
Sei in un atrio arredato con stile.
Vedo una porta blindata.

Cosa devo fare ? APRI LA PORTA
```

(tanto per passare il tempo)

```
Non hai la chiave.
Sei in un atrio arredato con stile.
Vedo una porta blindata.

[[[[ BOOM! ]]]]

Cosa devo fare ? B

Sei in una cantina ammuffita.
Vedo una chiave di sicurezza.
Vedo una cassaforte sventrata.
Vedo un cumulo di macerie.
```

Come dicevo, sarebbe più logico far cercare tra le macerie per trovare la chiave.

A proposito, se volete che una certa parola venga ignorata, come il `TRA` nella frase `CERCA TRA LE MACERIE`, basta metterla nel dizionario con codice 7.

Ho finito con l'agente segreto ed i suoi microfilm. Il prossimo capitolo mostra alcune comodità del Modulo Base che non ho avuto modo di mostrarvi nel corso di questa avventura.

CAPITOLO 11

TRUCCHI E COMODITÀ VARIE

Ovvero: come sfruttare il Modulo Base per risparmiare lavoro, come risolvere problemi strani e come aggiungere effetti speciali alle vostre avventure.

Nella costruzione di "Operazione Zanna Bianca", non ho sfruttato a fondo le possibilità offerte dal Modulo Base.

Colmo subito la lacuna, illustrandovi varie cose interessanti (e riassumendone alcune già viste di passata nei precedenti capitoli).

Iniziamo dalle subroutine di uso generale. Ne avete già usate due:

- `GOSUB 1500` stampa "Premi <spazio> per continuare" ed attende che il giocatore prema effettivamente la barra spaziatrice. È utile per evitare che una scritta troppo lunga scorra fuori dal video, o per aggiungere un elemento di suspense alla vicenda.

- `GOSUB 1600` stampa la domanda contenuta in A$, ed accetta come risposta soltanto `SI` o `NO` (o comunque parole che inizino con `S` o `N`. Al ritorno, la variabile logica A è vera se la risposta era `SI`, falsa se era `NO`.

Questa subroutine, invece, non la conoscevate ancora:

- `GOSUB 1700` fa una pausa di attesa, con una durata che dipende da A. Per la precisione, la pausa dura circa A secondi. Ad esempio:

```
A=5:GOSUB 1700
```

fa una pausa di 5 secondi. Sono ammessi anche valori decimali e minori di 1.

Se la subroutine 1700 viene nuovamente chiamata senza aver cambiato il valore di A, ripete ovviamente lo stesso ritardo dell'ultima volta.

Questo vale solo all'interno delle vostre routine di azione, dato che il Modulo Base cambia il contenuto di A in molte occasioni.
Una nota: le pause di attesa possono aggiungere suspense e coinvolgere il giocatore, ma possono anche annoiarlo se sono usate spesso o a sproposito.

In particolare, non mettete mai pause nell'introduzione, o comunque quando volete solo dare il tempo di leggere qualcosa. C'è chi legge velocemente ed è infastidito dalle pause (io, ad esempio). Usate invece il `GOSUB 1500` (premi <spazio> per continuare), che è fatto apposta.

Se trovate comodo aggiungere nuove subroutine di uso generale, che impiegate nelle vostre avventure, potete metterle nello spazio appositamente riservato nel Modulo Base, facendole iniziare alle linee 1800, 1900, 2000...

Ricordate di non fare mai un GOTO o GOSUB ad una linea di soli REM, per l'ottimo motivo che vi potrà capitare di togliere i REM per recuperare memoria.

Le variabili

Molte azioni danno un risultato diverso a seconda degli eventi accaduti in precedenza.

Ad esempio, CARICA LA SVEGLIA risponde "Già fatto" se la sveglia è già stata caricata, altrimenti la carica. Per sapere se la sveglia è carica, basta guardare se il timer 2 è in funzione.

In altri casi, l'informazione necessaria si ricava controllando se un certo oggetto è in un certo luogo (es. Porta chiusa/Porta aperta).

Ci sono però situazioni in cui non è possibile sapere se un'azione è stata compiuta o meno, ad esempio se è stata pronunciata una certa parola magica nel tempio di Zot. In questo caso, occorre un posto dove memorizzare l'informazione riguardante l'evento.

Il Modulo Base dispone di dieci variabili apposite, che vengono salvate insieme alla situazione dell'avventura e possono essere liberamente usate per conservare informazioni:

```
V1, V2, V3, V4, V5, V6, V7, V8, V9, V0
```

Una variabile può essere usata, ad esempio, per ricordare lo stato di un'azione complessa, che richiede tre o quattro azioni per essere completata.

Prendiamo ad esempio uno SCAVA che va ripetuto quattro o cinque volte (un'azione da ripetere è uno dei trucchi classici ma sempre validi):

```
9000 IF V3=0 THEN PRINT "Hai intaccato il terreno.":
  V3=1:RETURN
9010 IF V3=1 THEN PRINT "Hai fatto una piccola buca.":
  V3=2:RETURN
9020 IF V3=2 THEN PRINT "La terra e' umida.":V3=3:RETURN
9030 IF V3=3 THEN PRINT "Hai scavato un pozzo.":V3=4:RETURN
9040 PRINT "Non c'e' motivo di scavare ancora.":RETURN
```

Nella linea 9030, sarà anche il caso di far apparire il pozzo scavato.

La variabile V3, come dicevo, contiene lo "stato di avanzamento" dei lavori di scavo.

Dieci variabili possono sembrare poche, ma sono spesso sufficienti anche per avventure medie, dato che molte informazioni sono disponibili osservando il luogo in cui si trovano i vari oggetti.

Load e save

Le routine di registrazione e rilettura della situazione su disco sono piuttosto primitive, ma lo scopo di questo libro è di insegnarvi come scrivere avventure, non come si usa il DOS.

Se vi sentite in grado di farlo, potete modificarle per fare in modo che si possano registrare più situazioni, con nomi o numeri di identificazione differenti.

Attenti però alle trappole: dovete verificare che la situazione richiesta sia effettivamente su disco, e non lasciare che il programma si fermi se non la trova. Dovete tener conto della possibilità che non ci sia più spazio sul dischetto quando si fa SAVE, che il nome battuto non sia valido, ecc.

Non è difficilissimo, ma non è nemmeno semplice come sembra: come al solito, le rifiniture portano via più lavoro di tutto il resto.

Out of memory

Ahimé, sono certo che prima o poi vi capiterà. Di finire la memoria, intendo. Se scrivete un'avventura di una certa dimensione, finirete con il trovarvi in una situazione molto sgradevole: il programma sta tranquillamente in memoria, ma al RUN si ferma lamentando l'insufficienza della memoria disponibile.
Infatti, il programma richiede un certo spazio extra per le variabili numeriche e per i "puntatori" alle variabili stringa (dizionario, descrizioni, e nomi degli oggetti). Se questo manca, non c'è niente da fare.

Beh, non siamo così pessimisti: qualcosa da fare c'è sempre, ad esempio:

- Togliere tutti i REM dal Modulo Base. Ci sono programmi che lo fanno in modo automatico.
- Togliere le routine di azione non usate (difficilmente ne userete 160).
- Togliere le chiamate alle suddette routine di azione, nelle linee intorno alla 2600, ma solo se avete un'idea ben chiara di quello che state facendo.
- Riunire quante più istruzioni possibile sulla medesima linea di programma.
- Ridurre le dimensioni degli array (linee 1060-1090) allo stretto necessario. (al contrario, potrà capitarvi di doverle aumentare se non bastano per la vostra avventura).

Alla fine di tutto questo, avrete recuperato un bel po' di memoria. Se ne volete dell'altra, avete la scelta tra usare messaggi più corti e trasferire i messaggi in un file su disco, scrivendo una routine che vada a leggerli quando necessario.

Ma a questo punto sarete sufficientemente esperti da poter passare all'uso del nuovo Modulo Base, che libera gran parte della memoria perché legge descrizioni e messaggi da disco (vddi Appendice C).

A proposito, eliminando i controlli dalla linea 1170 alla 1220 (cancellando semplicemente le linee) si risparmia ancora un poco di spazio e si riduce l'attesa iniziale. È però il caso di farlo solo quando il gioco è a posto, altrimenti si rischiano inconvenienti strani se qualche parola o azione non è in ordine (ad esempio: introducendo un'azione, non ne funziona più un'altra).

Effetti speciali

Un'avventura può essere resa più interessante aggiungendo effetti sonori e grafici. Per quanto riguarda i primi, potete sfruttare le notevoli capacità del vostro BASIC: in particolare, l'istruzione PLAY consente di inserire stacchetti o brevi musiche sotto forma di stringhe.

Passando alla grafica, è chiaro che le descrizioni potrebbero essere sostituite da disegni, anche se non è detto che l'avventura ci guadagni: la fantasia del giocatore, se ben stimolata, immaginerà una scena migliore di qualunque disegno. In ogni caso, potete preparare i disegni con uno degli appositi programmi, e richiamarli al momento opportuno.
Alcuni effetti sono però ottenibili anche senza fare uso del disco. In molti casi, un accorto uso di `LINE`, `CIRCLE`, e simili può essere sufficiente per creare qualcosa di inatteso.

Un solo suggerimento: potreste disegnare gli oggetti (in risposta a `GUARDA IL ...`) usando l'istruzione `DRAW`. Il disegno di ciascun oggetto potrebbe essere contenuto in una stringa che viene letta dai DATA all'inizio del gioco, ad esempio nell'array `DO$` (disegni oggetti), insieme alle altre caratteristiche degli oggetti stessi.

Seguite il buon esempio

Se volete imparare un po' di sistemi & trucchetti vari per sfruttare le possibilità offerte dal Modulo Base, studiatevi le tre avventure che avete giocato. Troverete spunti sul modo di affrontare problemi che potrebbe capitare anche a voi di dover risolvere.

In particolare, ne "L'astronave condannata", meritano attenzione:

- Le azioni `PRENDI` e `LASCIA`, con particolare riferimento a casco, tuta, camice e spazio interplanetario (senz'aria).
- L'azione `GUARDA L'INDICATORE`.
- Il meccanismo del compartimento stagno.
- Il controllo del reattore.

Anche ne "L'anello di Lucrezia Borgia" potete trovare idee e suggerimenti, ad esempio osservando come funzionano:

- L'orso che insegue (occhio all'azione 1, quella delle direzioni).
- L'effetto dell'elleboro, con la ciotola che si rovescia.
- La sequenza della preghiera, che usa una variabile.
- La battaglia, esempio di uso di semplici effetti speciali.
- La pianura (osservando mappa e azioni).

"L'Apprendista Stregone" è utile invece per rendersi conto delle possibilità offerte dal nuovo Modulo Base.

Non vi do consigli su come trovare le azioni citate nei programmi: ormai siete esperti.

APPENDICE A

LISTATO DEL MODULO BASE

Il Modulo Base è scritto in BASIC il più possibile semplice e lineare, per consentirvi di esaminare il funzionamento del programma. Mancano molte finiture tipiche dei programmi commerciali, che avrebbero aggiunto complessità mascherando l'essenziale.

Con un po' di buona volontà, il Modulo Base può essere adattato alla produzione commerciale di giochi di avventura. Vi segnalo alcuni punti del programma particolarmente suscettibili di miglioramento.

Per prima cosa, l'input (linea 800) dovrebbe essere compito di un'apposita routine, che impedisca all'utente di andare a spasso per il video con il cursore e di fermare il programma con CTRL-C.

Sarebbe anche il caso di accettare caratteri sia maiuscoli che minuscoli.

Il programma andrebbe compilato, per due motivi: aumentarne la velocità di esecuzione e rendere meno facile la lettura delle risposte. Il giocatore troppo curioso non deve trovare facilmente una scorciatoia.

Si potrebbe riscrivere il tutto in un BASIC più pulito e strutturato, come il Turbo Basic, con il vantaggio di poter chiamare routine ed azioni per nome.
E poi, di migliorie ce ne starebbero tante altre... il perfezionismo non ha mai fine!

Comunque, ecco il listato:

```
100 REM ## Avventura: Interprete e modulo base ##
110 REM ##         di Enrico Colombini         ##
120 REM ##  (c) Dinosoft e Jackson 1985,1988   ##
122 REM ##  (c) Jackson Libri - Gr.Futura 1992 ##
124 REM ##  (c) Enrico Colombini 1999          ##
130 REM
140 GOTO 710:REM MAIN
150 REM
160 REM - CERCA P$ IN DIZIONARIO, C=CODICE (0 SE ASSENTE) -
170 REM
180 I=1:F=FD
190 A=INT((I+F)/2):A$=DZ$(A):IF P$=A$ THEN C=DZ(A):GOTO 240
200 IF P$>A$ THEN I=A+1
210 IF P$<A$ THEN F=A-1
220 IF I<=F THEN 190
230 C=0
240 RETURN
250 REM
260 REM - ESTRAE P$ DA IN$[IN], TROVA CODICE C,
  SALTA ARTICOLI -
270 REM
```

```
280 C=0
290 C$=MID$(IN$,IN,1):IF C$=" " OR C$="'" THEN IN=IN+1:
  GOTO 290
300 IF IN>LI THEN P$="":GOTO 360
310 A=IN:REM INIZIO
320 C$=MID$(IN$,IN,1)
330 IF C$<>" " AND C$<>"'" AND IN<=LI THEN IN=IN+1:GOTO 320
340 P$=MID$(IN$,A,IN-A):GOSUB 180:REM CERCA
350 IF C=7 THEN 280:REM ARTICOLO
360 RETURN
370 REM
380 REM - CERCA AZIONE A, A=AZIONE (0 SE NON TROVATA) -
390 REM
400 I=1:F=FA:N=A
410 A=INT((I+F)/2):M=CA(A):IF N=M THEN A=AZ%(A):GOTO 460
420 IF N>M THEN I=A+1
430 IF N<M THEN F=A-1
440 IF I<=F THEN 410
450 A=0
460 RETURN
470 REM
480 REM - ESEGUE AZIONE A, A=0 SE NON TROVATA -
490 REM
500 GOSUB 400:IF A=0 THEN 550
510 IF C2=0 OR A>0 THEN 530:REM SOLO VERBO O NO TEST
520 A=-A:IF OG=0 THEN PRINT "- Qui non ne vedo.":GOTO 540
530 GOSUB 2500:REM ESEGUE
540 A=1
550 RETURN
560 REM
570 REM - ELENCA OGGETTI IN LUOGO L, CON PREFISSO P$,
  A=0 SE NESSUNO -
580 REM
590 A=0:FOR I=1 TO FO
600 IF ABS(L(I))=L THEN PRINT P$;OG$(I);".":A=1
610 NEXT:RETURN
620 REM
630 REM - OG=INDICE OGGETTO C2, 0 SE NON PRES. O TRASP. -
640 REM
650 OG=0:FOR I=1 TO FO
660 IF CO%(I)=C2 THEN IF ABS(L(I))=LU OR L(I)=0 THEN
  OG=I:I=FO
670 NEXT:RETURN
680 REM
690 REM - MAIN (PARSER) -
700 REM
710 GOSUB 1060:REM INIT
720 GOSUB 4500:REM INTRODUZIONE
730 GOSUB 1500:REM <SP>
740 REM
750 REM CICLO DI GIOCO:
760 PRINT:PRINT "Sei ";DE$(LU);".":REM DESCRIZIONE
770 L=LU:P$="Vedo ":GOSUB 590:REM OGGETTI
780 GOSUB 970:REM TEMPO
790 PRINT:PRINT
800 IN$="":INPUT "Cosa devo fare ";IN$:IF IN$="" THEN 800
805 PRINT:IF ASC(LEFT$(IN$,1))>96 THEN PRINT"- Usa le
  MAIUSCOLE.":GOTO 760
810 LI=LEN(IN$):IN=1:GOSUB 280:P1$=P$:C1=C
```

```
820 IF P1$="" THEN PRINT "- Beh ?":GOTO 760
830 IF RIGHT$(P1$,2)="RE" THEN PRINT "- Dammi del tu, per
  favore.":GOTO 760
840 IF C1=0 AND P1$<>"" THEN PRINT "- Non conosco il verbo
  '"P1$"'.":GOTO 760
850 GOSUB 280:P2$=P$:C2=C:IF C2>0 AND C2<7 THEN 850: REM
  NO ARTICOLI
860 IF C2=0 AND P2$<>"" THEN PRINT "- Non conosco la parola
  '"P2$"'.":GOTO 760
870 IF C2<>0 THEN GOSUB 650:REM OG=INDICE
880 N1=LU*10000:N2=C1*100
890 A=N1+N2+C2:GOSUB 500:IF A GOTO 760:REM VERBO+NOME IN LU
900 IF C2<>0 THEN A=N1+N2+99:GOSUB 500:IF A GOTO 760: REM
  VERBO+X IN LU
910 A=N2+C2:GOSUB 500:IF A GOTO 760:REM VERBO+NOME GENERICO
920 IF C2<>0 THEN A=N2+99:GOSUB 500:IF A GOTO 760: REM
  VERBO+X GENERICO
930 PRINT "- Non capisco.":GOTO 760
940 REM
950 REM - TEMPO -
960 REM
970 IF T1 THEN T1=T1-1:GOSUB 3000
980 IF T2 THEN T2=T2-1:GOSUB 3200
990 IF T3 THEN T3=T3-1:GOSUB 3400
1000 IF T4 THEN T4=T4-1:GOSUB 3600
1010 IF T5 THEN T5=T5-1:GOSUB 3800
1020 RETURN
1030 REM
1040 REM - INIT -
1050 REM
1060 DIM DZ$(150),DZ(150):REM DIZIONARIO
1070 DIM DE$(98),DI$(98):REM MAPPA
1080 DIM CA(160),AZ%(160):REM AZIONI
1090 DIM OG$(50),CO%(50),L(50):REM OGGETTI
1100 F$="SITUAZ.SIT":REM FILE SITUAZIONE
1110 REM FD,FM,FA,FO=0
1120 KEY OFF:PRINT:PRINT "Un attimo di pazienza...";
1130 READ A$:IF A$<>"FD" THEN FD=FD+1:DZ$(FD)=A$:READ
  DZ(FD):GOTO 1130
1140 READ A$:IF A$<>"FM" THEN FM=FM+1:DE$(FM)=A$:READ
  DI$(FM):GOTO 1140
1150 READ A$:IF A$<>"FA" THEN FA=FA+1:CA(FA)=VAL(A$):
  READ AZ%(FA):GOTO 1150
1160 READ A$:IF A$<>"FO" THEN FO=FO+1:OG$(FO)=A$:
  READ CO%(FO),L(FO):GOTO 1160
1170 FOR I=2 TO FD:REM --- CONTROLLO DIZIONARIO
  (ELIMINABILE) ---
1180 IF DZ$(I)<=DZ$(I-1) THEN PRINT "Errore: "DZ$(I)
  " <= "DZ$(I-1):END
1190 NEXT
1200 FOR I=2 TO FA:REM --- CONTROLLO AZIONI
  (ELIMINABILE) ---
1210 IF CA(I)<=CA(I-1) THEN PRINT "Errore: "CA(I)" <=
  "CA(I-1):END
1220 NEXT
1230 PRINT:PRINT:RETURN
1470 REM
1480 REM - <SP> PER CONTINUARE -
1490 REM
```

```
1500 PRINT:PRINT "- Premi <spazio> per continuare -";
1510 A$=INPUT$(1):IF A$<>" " THEN 1510
1520 PRINT:PRINT:RETURN
1570 REM
1580 REM - DOMANDA A$, A=VERO SE SI' -
1590 REM
1600 PRINT A$;:INPUT " ";B$:B$=LEFT$(B$,1)
1610 IF B$<>"S" AND B$<>"N" THEN 1600
1620 A=(B$="S"):RETURN
1670 REM
1680 REM - PAUSA A SECONDI CIRCA -
1690 REM
1700 FOR I=1 TO 20*A:PLAY"p64.":NEXT:RETURN
1710 REM
1720 REM * (METTERE QUI ALTRE SUBROUTINE DI USO GENERALE) *
2470 REM
2480 REM - ESEGUE AZIONE A -
2490 REM
2500 ON INT((A-1)/20) GOTO 2510,2530,2550,2570,2590,
  2610,2630,2650
2510 ON A GOTO 5000,5200,5400,5600,5800,
  6000,6200,6400,6600,6800
2520 ON A-10 GOTO 7000,7200,7400,7600,7800,
  8000,8200,8400,8600,8800
2530 ON A-20 GOTO 9000,9200,9400,9600,9800,
  10000,10200,10400,10600,10800
2540 ON A-30 GOTO 11000,11200,11400,11600,11800,
  12000,12200,12400,12600,12800
2550 ON A-40 GOTO 13000,13200,13400,13600,13800,
  14000,14200,14400,14600,14800
2560 ON A-50 GOTO 15000,15200,15400,15600,15800,
  16000,16200,16400,16600,16800
2570 ON A-60 GOTO 17000,17200,17400,17600,17800,
  18000,18200,18400,18600,18800
2580 ON A-70 GOTO 19000,19200,19400,19600,19800,
  20000,20200,20400,20600,20800
2590 ON A-80 GOTO 21000,21200,21400,21600,21800,
  22000,22200,22400,22600,22800
2600 ON A-90 GOTO 23000,23200,23400,23600,23800,
  24000,24200,24400,24600,24800
2610 ON A-100 GOTO 25000,25200,25400,25600,25800,
  26000,26200,26400,26600,26800
2620 ON A-110 GOTO 27000,27200,27400,27600,27800,
  28000,28200,28400,28600,28800
2630 ON A-120 GOTO 29000,29200,29400,29600,29800,
  30000,30200,30400,30600,30800
2640 ON A-130 GOTO 31000,31200,31400,31600,31800,
  32000,32200,32400,32600,32800
2650 ON A-140 GOTO 33000,33200,33400,33600,33800,
  34000,34200,34400,34600,34800
2660 ON A-150 GOTO 35000,35200,35400,35600,35800,
  36000,36200,36400,36600,36800
2670 PRINT A" ???":RETURN:REM PER SICUREZZA
2920 REM
2930 REM
2940 REM ### FINE INTERPRETE, INIZIO AVVENTURA ###
2950 REM
2960 REM
2970 REM - TIMER -
```

```
2980 REM
2990 REM T1:
3000 PRINT:PRINT "T1":RETURN
3190 REM T2:
3200 PRINT:PRINT "T2":RETURN
3390 REM T3:
3400 PRINT:PRINT "T3":RETURN
3590 REM T4:
3600 PRINT:PRINT "T4":RETURN
3790 REM T5:
3800 PRINT:PRINT "T5":RETURN
3970 REM
3980 REM - MORTO -
3990 REM
4000 GOSUB 1500:REM <SP>
4010 REM --- METTERE QUI IL FINALE NEGATIVO ---
4450 A$="Vuoi giocare ancora":GOSUB 1600:IF(A) THEN RUN:
  REM RIPARTE
4460 PRINT:PRINT:PRINT "Ciao!":PRINT:END
4470 REM
4480 REM - INTRODUZIONE -
4490 REM
4500 CLS:PRINT:PRINT:PRINT
4510 REM --- METTERE QUI INTRODUZIONE AVVENTURA ---
4520 REM --- RICORDARSI LU=LUOGO INIZIALE ---
4950 LU=1:PRINT:RETURN
4960 REM
4970 REM - AZIONI: -
4980 REM
4990 REM 1: (DIREZIONI)
5000 A=VAL( MID$(DI$(LU),2*C1-1,2))
5010 IF A=0 THEN PRINT "- Di li' non puoi andare.":RETURN
5020 LU=A:RETURN
5190 REM 2: (PRENDI)
5200 IF L(OG)=0 THEN PRINT "- Gia' fatto.":RETURN
5210 IF L(OG)<0 THEN PRINT "- Non e' possibile.":RETURN
5220 L(OG)=0:PRINT "Fatto.":RETURN
5390 REM 3: (LASCIA)
5400 IF OG=0 OR L(OG)<>0 THEN PRINT "- Non ce l'hai.":
  RETURN
5410 L(OG)=LU:PRINT "Fatto.":RETURN
5590 REM 4: (GUARDA)
5600 PRINT "Non noto nulla di particolare.":RETURN
5790 REM 5: (SAVE)
5800 OPEN F$ FOR OUTPUT AS #1
5810 FOR I=1 TO FO:PRINT #1,L(I):NEXT
5820 WRITE #1,LU,T1,T2,T3,T4,T5,
  V1,V2,V3,V4,V5,V6,V7,V8,V9,V0
5830 CLOSE #1:RETURN
5990 REM 6: (LOAD)
6000 OPEN F$ FOR INPUT AS #1
6010 FOR I=1 TO FO:INPUT #1,L(I):NEXT
6020 INPUT #1,LU,T1,T2,T3,T4,T5,
  V1,V2,V3,V4,V5,V6,V7,V8,V9,V0
6030 CLOSE #1:RETURN
6190 REM 7: (COSA)
6200 PRINT "Possiedi:":PRINT:L=0:P$="- ":GOSUB 590
6210 IF A=0 THEN PRINT "- Un bel nulla."
6220 RETURN
```

```
6390 REM 8:
6400 PRINT 8:RETURN
6590 REM 9:
6600 PRINT 9:RETURN
6790 REM 10:
6800 PRINT 10:RETURN
....
```

(continua così, con le routine di azione a passo 200, fino alla 34800)

```
.....
34990 REM 151:
35000 PRINT 151:RETURN
35190 REM 152:
35200 PRINT 152:RETURN
35390 REM 153:
35400 PRINT 153:RETURN
35590 REM 154:
35600 PRINT 154:RETURN
35790 REM 155:
35800 PRINT 155:RETURN
35990 REM 156:
36000 PRINT 156:RETURN
36190 REM 157:
36200 PRINT 157:RETURN
36390 REM 158:
36400 PRINT 158:RETURN
36590 REM 159:
36600 PRINT 159:RETURN
36790 REM 160:
36800 PRINT 160:RETURN
39970 REM
39980 REM - DIZIONARIO:
39990 REM
40000 DATA "?",13,"A",5,"AGLI",7,"AL",7,"ALL",7
40010 DATA "ALLA",7,"ALLE",7,"ALLO",7,"ALTO",5
40020 DATA "B",6,"BASSO",6
40030 DATA "COL",7,"CON",7,"COSA",13
40040 DATA "E",3,"EST",3
40050 DATA "GLI",7,"GUARDA",10
40060 DATA "I",7,"IL",7,"INVENTARIO",13
40070 DATA "L",7,"LA",7,"LASCIA",9
40080 DATA "LE",7,"LO",7,"LOAD",12
40090 DATA "N",1,"NORD",1
40100 DATA "O",4,"OVEST",4
40110 DATA "POSA",9,"PRENDI",8
40120 DATA "RIPRENDI",12
40130 DATA "S",2,"SALI",5,"SALVA",11
40140 DATA "SAVE",11,"SCENDI",6,"SUD",2
40150 DATA "UN",7,"UNA",7,"UNO",7
40160 DATA "W",4
40170 DATA "FD":REM --- RICORDARSI L'ORDINE ALFABETICO!!! -
44970 REM
44980 REM - MAPPA:
44990 REM
45000 DATA "FM": REM --- LUOGHI E MAPPA, DAL LUOGO N.1 ---
49970 REM
49980 REM - AZIONI:
49990 REM
```

```
50000 DATA 100,1,200,1,300,1,400,1,500,1,600,1,899,-2,999,3
50010 DATA 1000,4,1099,-4,1100,5,1200,6,1300,7
50020 DATA "FA":REM --- RICORDARSI L'ORDINE NUMERICO!!! ---
54970 REM
54980 REM - OGGETTI:
54990 REM
55000 DATA "FO":REM --- DESCRIZIONI E DATI OGGETTI, DAL N.1
```

APPENDICE B

USO DEL MODULO BASE: RIASSUNTO

Questa è una guida rapida di riferimento per l'uso del Modulo Base. Riassume i punti essenziali di quanto visto nei vari capitoli a proposito di mappa, dizionario, oggetti, azioni e routine varie.

Avete seguito i capitoli del libro, diventando così esperti autori di avventure? Molto bene. Ora avete solo bisogno di ricordarvi come si usa il Modulo Base, che numeri bisogna scrivere e dove metterli. È tutto qui di seguito.

Mappa:

Inserire descrizioni luoghi e direzioni come DATA dalla linea 45000 in poi, in ordine di numero di luogo (dal n.1) e terminando con "FM".

Le direzioni si indicano con 12 cifre: le prime due indicano il numero del luogo adiacente a Nord, le successive quello a Sud, poi ad Est, Ovest, Alto, Basso.

Se lo spostamento non è possibile, mettere 00 (doppio zero) come numero del luogo.

Esempio:

```
45000 DATA "in fondo al pozzo","000000002400"
```

Si può andare solo in alto, e si finisce nel luogo 24 (il 24º dell'elenco).

Dizionario:

Inserire parole e codici come DATA dalla linea 40000 in poi, ricordandosi di mantenere l'ordine alfabetico e terminare con "FD".
Esempio:

```
40000 DATA "?",13,"A",5,"ACCENDI",35,"AGLI",7
```

La nuova parola "ACCENDI", con codice 35, è stata inserita in mezzo a quelle già presenti nel dizionario, in ordine alfabetico.

Oggetti:

Inserire descrizione, codice parola di riferimento e luogo iniziale come DATA a partire dalla linea 55000, in ordine di numero di oggetto (dal n.1) e terminando con "FO".

Esempio:

```
55000 DATA "un libro di avventure",42,13
```

La parola a cui l'oggetto risponde (probabilmente "LIBRO") si trova nel dizionario con codice 42. Il libro si trova, all'inizio del gioco, nel luogo 13.

Se un oggetto non è prendibile, va indicato con codice negativo:

```
55010 DATA "un masso di granito rosa",44,-13
```

Se l'oggetto non è in gioco, si indica -99 (il Limbo) come numero di luogo:

```
55020 DATA "un brontosauro sonnolento",52,-99
```

Sel'oggetto si deve trovare, all'inizio del gioco, in possesso dell'avventuriero, si indica il luogo zero:

```
55030 DATA "un orologio scarico",55,0
```

Il numero di un oggetto corrisponde alla sua posizione nell'elenco dei DATA. Riguardando gli esempi fatti, il libro sarebbe l'oggetto 1, il masso l'oggetto 2, il brontosauro l'oggetto 3, e l'orologio l'oggetto 4.

Azioni:

Inserire codice frase e numero azione come DATA dalla linea 50000 in poi, ricordandosi di <u>mantenere l'ordine numerico e terminare con "FA".</u>

Il codice di frase è un numero composto da:

- Numero del luogo (2 cifre).
- Codice della prima parola (2 cifre).
- Codice della seconda parola (2 cifre).

Gli zeri iniziali possono poi essere eliminati.

Con frasi di una sola parola, per il codice di frase sono possibili queste due combinazioni (in ordine di precedenza):

PRENDI	(nel luogo 13)	131000	(luogo 13/PRENDI/nessuna parola)
PRENDI	(ovunque)	001000	(ovunque/PRENDI/nessuna parola)

Con frasi di due parole (articoli e preposizioni non contano), ci sono quattro possibili combinazioni, sempre in ordine di precedenza:

PRENDI LIBRO	(luogo 13)	131042	(luogo 13/PRENDI/LIBRO)
PRENDI X	(luogo 13)	131099	(luogo 13/PRENDI/qualunque parola)
PRENDI LIBRO	(ovunque)	001042	(ovunque/PRENDI/LIBRO)
PRENDI X	(ovunque)	001099	(ovunque/PRENDI/qualunque parola)

Al codice di frase segue il numero dell'azione da eseguire.

Se questo numero è negativo, l'azione è eseguita soltanto se la seconda parola della frase indica un oggetto presente o trasportato. Esempio:

```
50010 DATA 1000,4,1042,-12,1099,-4,1100,5,1200,6,1300,7
```

Il nuovo codice di frase 1042, è stato inserito in mezzo a quelli già presenti nella tavola, in ordine numerico, seguito dal numero dell'azione da eseguire (-12).

Se 10 è il codice di "GUARDA" nel dizionario, e 42 è il codice di "LIBRO", la frase "GUARDA IL LIBRO" fa eseguire la subroutine di azione numero 12 ma solo se l'oggetto citato (il libro) è presente o trasportato (a causa del segno meno davanti al 12).

Subroutine di azione:

Una frase che coincide con un codice di azione presente nella tavola causa l'esecuzione della corrispondente subroutine di azione.

La subroutine di azione numero 1 inizia alla linea 5000, la numero 2 alla 5200, e così via di 200 in 200 fino alla subroutine numero 160, che si trova alla linea 36800. I numeri di linea non vanno modificati.

Ogni subroutine di azione è composta di normali istruzioni BASIC e deve terminare con l'istruzione RETURN.

Timer:

I cinque timer sono controllati dalle variabili T1, T2, T3, T4, T5.

Se una variabile di timer è a zero, il timer è fermo.

Per far partire un timer, si mette un numero nella variabile corrispondente. Ad esempio, per far partire il timer 2:

```
T2=3
```

Una volta avviato, il timer conta all'indietro finché non arriva a zero, poi si ferma.

Se un timer è in funzione, ad ogni mossa chiama la corrispondente routine.

La routine del timer 1 inizia alla linea 3000.
La routine del timer 2 inizia alla linea 3200.
La routine del timer 3 inizia alla linea 3400.
La routine del timer 4 inizia alla linea 3600.
La routine del timer 5 inizia alla linea 3800.

Variabili principali

Le informazioni sull'avventura sono contenute in alcune variabili fondamentali, che possono essere modificate a piacimento:

LU	Numero del luogo corrente.
OG	Numero dell'oggetto citato nella frase (0 se non è un oggetto presente o trasportato).
L(n)	Luogo dell'oggetto numero n (0=trasp., -99=Limbo), negativo se l'oggetto non è prendibile.
V0..V9	Variabili libere, usabili a piacere.
T1..T5	Timer, partono mettendoci un numero maggiore di zero.

Esempio di azione numero 12 (linea 7200):

```
7200 IF L(1) <> 0 THEN PRINT "Non hai il libro.": RETURN
```

```
7210 PRINT "Leggi solo 'FUOCO', poi il libro scompare."
7220 L(1) = -99 : RETURN : REM Mette il libro nel Limbo
```

L'oggetto numero 1 (il libro) è quello elencato per primo nei DATA alla linea 55000. L(1) è dunque il luogo in cui si trova il libro.
Occorre fare attenzione agli oggetti non prendibili, il cui numero di luogo è e deve restare negativo. Modificandolo, l'oggetto diventa prendibile, ma occorre modificare le routine di `SAVE e LOAD` per registrare anche questo numero.

Indirizzi utili (numeri di linea):

`GOSUB 1500`	- Premi <spazio> per continuare:
`GOSUB 1600`	Stampa la domanda contenuta in A$, torna con A=vero se la risposta è Sì, falso se è No.
`GOSUB 1700`	Aspetta A secondi circa.
`GOSUB 3000`	Subroutine di timer 1 (eseguita automaticamente ad ogni turno se il timer 1 è attivo). Le successive sono alle linee 3200 (T2), 3400 (T3), 3600 (T4) e 3800 (T5)
`GOSUB 5000`	Subroutine di azione 1 (eseguita automaticamente), le successive sono alla 5200 (2), 5400 (3), ecc.

Nella pagina seguente sono riportate gran parte delle variabili del programma che hanno significato ai fini dell'avventura.

Mappa:

`FM`	Numero luoghi in mappa.
`DE$(1..FM)`	Descrizioni luoghi, in ordine di numero.
`DI$(1..FM)`	Collegamenti del luogo descritto nel corrispondente elemento di `DE$()` con i sei luoghi adiacenti (N/S/E/O/A/B).

Dizionario:

`FD`	Numero vocaboli in dizionario.
`DZ$(1..FD)`	Vocaboli, in ordine alfabetico.
`DZ%(1..FD)`	Codici dei vocaboli corrispondenti in `DZ$()`.

Azioni:

`FA`	Numero azioni nella tavola.
`CA(1..FA)`	Codici di azione, in ordine numerico.
`AZ%(1..FA)`	Numero azione corrispondente da eseguire, negativo se è richiesta la presenza dell'oggetto citato.

Oggetti:

`FO`	Numero oggetti presenti.
`OG$(1..FO)`	Descrizioni oggetti, in ordine di numero oggetto.
`CO%(1..FO)`	Codice nel dizionario dell'oggetto descritto nel corrispondente elemento di `OG$()`.
`L(1..FO)`	Luogo dell'oggetto corrispondente: `1..FM` luogo indicato, prendibile. `-1..-FM` luogo indicato, non prendibile. `0` trasportato dall'avventuriero. `-99` limbo (fuori gioco).

Varie:

`LU`	Luogo corrente dell'avventuriero.
`OG`	Numero dell'oggetto citato come seconda parola, zero se non è un oggetto presente o trasportato.
`P1$,P2$`	Prima e seconda parola valida della frase.
`C1,C2`	Codici nel dizionario di `P1$` e `P2$`

APPENDICE C

IL NUOVO MODULO BASE

Il nuovo Modulo Base permette di creare avventure più complesse, e più divertenti da giocare grazie alle lunghe descrizioni che non rubano memoria al BASIC.

Tra i lettori della prima edizione di questo libro, molti hanno iniziato a scrivere proprie avventure. Chi ha proseguito su questa strada si è presto reso conto dei limiti del Modulo Base: poche parole nel dizionario, pochi luoghi, e soprattutto poco spazio per le descrizioni.
D'altra parte il Modulo Base è nato con intento didattico, e non come strumento professionale.

Nel preparare questa seconda edizione, ho pensato di aggiungere qualcosa anche per gli esperti: una versione potenziata del Modulo Base, adatta anche per creare avventure di discrete dimensioni. Ecco le principali migliorie introdotte:

- Codici delle parole e numeri dei luoghi a 3 cifre, anziché a 2.

- Descrizioni e risposte prelevabili da un file, senza i limiti del BASIC.

- Descrizioni lunghe e brevi per i luoghi.

- Oggetti invisibili.

- Uso del colore nelle scritte, per chi ha un monitor adatto.

Non poteva mancare un'avventura costruita con BASE2: "L'Apprendista Stregone" (ogni riferimento ai creatori di avventure è puramente casuale).

Il nuovo Modulo Base (versione 2.0) è nel file BASE2.BAS, e la relativa documentazione è sul disco stesso, nel file testo BASE2.DOC, anche per facilità di aggiornamento. È ben più facile modificare un dischetto che un libro, specialmente all'ultimo minuto.

Spero che chi vuol cimentarsi in grandi imprese sappia almeno leggere un file testo (se non altro con TYPE), come per l'appunto BASE2.DOC.
A proposito di innovazioni, i BASIC moderni (es. Turbo BASIC, QuickBasic) sono assai superiori al buon vecchio GWBASIC: niente numeri di linea, struttura più pulita, nomi per le routine, procedure con passaggio di parametri, ecc.

Mi sarebbe piaciuto scrivere il nuovo Modulo Base con uno di questi BASIC moderni, buttando i GOTO dalla finestra, ma non potevo lasciare a piedi tutti coloro che ancora usano il GWBASIC. Inoltre, sarebbe stato necessario un altro libro per spiegarne il modo d'uso.

Ho però cercato di scrivere il nuovo Modulo Base in modo che funzioni correttamente anche con il Turbo BASIC. Non è stato semplice, specialmente

per quanto riguarda i messaggi da disco: chi ha mai visto due BASIC compatibili scagli il primo cristallo di silicio.

Ho provato "L'astronave condannata" e "L'anello di Lucrezia Borgia" con il Turbo BASIC, e sembrano funzionare correttamente. Lo stesso dicasi per "L'Apprendista Stregone". Non ho però avuto il tempo di ripetere gli approfonditi collaudi anche sulla versione compilata, e perciò non posso garantirne la compatibilità al 100%.

Avrei ancora molte cose da dire, e non mi spiacerebbe scrivere altri strumenti (ed altre avventure). Se questo libro avrà una buona accoglienza, perché non pubblicare un Modulo Base per avventure grafiche, o una versione Pascal, o C, o Prolog? O un interprete per giochi di ruolo, o libri-game, o...

L'importante è imparare divertendosi !